

2017

## Procedural Historic Building Information Modelling (HBIM) For Recording and Documenting European Classical Architecture

Conor Dore

*Technological University Dublin, conor.dore@mydit.ie*

Follow this and additional works at: <https://arrow.tudublin.ie/builtloc>



Part of the [Construction Engineering and Management Commons](#)

---

### Recommended Citation

Dore, C. (2017) *Procedural Historic Building Information Modelling (HBIM) For Recording and Documenting European Classical Architecture*. Doctoral thesis, DIT, 2017.

This Theses, Ph.D is brought to you for free and open access by the Built Environment at ARROW@TU Dublin. It has been accepted for inclusion in Doctoral by an authorized administrator of ARROW@TU Dublin. For more information, please contact [yvonne.desmond@tudublin.ie](mailto:yvonne.desmond@tudublin.ie), [arrow.admin@tudublin.ie](mailto:arrow.admin@tudublin.ie), [brian.widdis@tudublin.ie](mailto:brian.widdis@tudublin.ie).



This work is licensed under a [Creative Commons Attribution-NonCommercial-Share Alike 3.0 License](#)

# Procedural Historic Building Information Modelling (HBIM)

*For Recording and Documenting European Classical  
Architecture*

Conor Dore, BSc.



Lead Supervisor: Dr. Maurice Murphy

Advisory Supervisor: Dr Eugene McGovern

School of Surveying and Construction Management  
Dublin Institute of Technology

**Doctor of Philosophy**

**August 2017**

## **DECLARATION**

I certify that this thesis which I now submit for examination for the award of Doctor of Philosophy, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. This thesis was prepared according to the regulations for graduate study by research of the Dublin Institute of Technology and has not been submitted in whole or in part for another award in any other third level institution. The work reported on in this thesis conforms to the principles and requirements of the DIT's guidelines for ethics in research. DIT has permission to keep, lend or copy this thesis in whole or in part, on condition that any such use of the material of the thesis be duly acknowledged.

Signature \_\_\_\_\_ Date \_\_\_\_\_

Candidate

## **ACKNOWLEDGEMENTS**

I would like to thank my lead supervisor Dr Maurice Murphy for his continuous support and assistance throughout this research. I would also like to thank my advisory supervisor Dr Eugene McGovern for his assistance with this project. I would like to thank the Dublin Institute of Technology and the Irish Research Council for the support and funding of this research. Finally, I would like to thank my family for their continuous support.



## ABSTRACT

*Procedural Historic Building Information Modelling (HBIM) is a new approach for modelling historic buildings which develops full building information models from remotely sensed data. HBIM consists of a novel library of reusable parametric objects, based on historic architectural data and a system for mapping these library objects to survey data. Using concepts from procedural modelling, a new set of rules and algorithms have been developed to automatically combine HBIM library objects and generate different building arrangements by altering parameters. This is a semi-automatic process where the required building structure and objects are first automatically generated and then refined to match survey data.*

*The encoding of architectural rules and proportions into procedural modelling rules helps to reduce the amount of further manual editing that is required. The ability to transfer survey data such as building footprints or cut-sections directly into a procedural modelling rule also greatly reduces the amount of further editing required. These capabilities of procedural modelling enable a more automated and efficient overall workflow for reconstructing BIM geometry from point cloud data.*

*This document outlines the research carried out to evaluate the suitability of a procedural modelling approach for improving the process of reconstructing building geometry from point clouds. To test this hypothesis, three procedural modelling prototypes were designed and implemented for BIM software. Quantitative accuracy testing and qualitative end-user scenario testing methods were used to evaluate the research hypothesis. The results obtained indicate that procedural modelling has potential for achieving more accurate, automated and easier generation of BIM geometry from point clouds.*

# TABLE OF CONTENTS

TABLE OF FIGURES.....	I
TABLE OF TABLES.....	XIII
LIST OF APPENDICES.....	XIV
GLOSSARY .....	XV

## Part I: *INTRODUCTION*

CHAPTER ONE: INTRODUCTION .....	2
1.1 BACKGROUND, MOTIVATION AND CONTEXT .....	2
1.2 RESEARCH PROBLEM .....	5
1.3 RESEARCH QUESTION .....	6
1.4 AIM AND OBJECTIVES OF RESEARCH .....	7
1.5 SCOPE OF RESEARCH .....	8
1.6 CONTRIBUTIONS .....	9
1.7 RESEARCH METHODOLOGY .....	10
1.8 PUBLICATIONS .....	11
1.9 FORMAT OF DOCUMENT .....	12
CHAPTER TWO: CURRENT STATE OF THE ART.....	14
2.1 INTRODUCTION .....	14
2.2 HERITAGE DOCUMENTATION STANDARDS .....	14
2.3 DATA COLLECTION AND PRE-PROCESSING TECHNIQUES.....	17
2.3.1 <i>Terrestrial Laser Scanning</i> .....	17
2.3.2 <i>Photogrammetry</i> .....	18
2.3.3 <i>Other Survey Techniques</i> .....	18

2.3.4	<i>Pre-Processing Laser and Image Data</i> .....	19
2.4	3D MODELLING CONCEPTS .....	20
2.5	AS-BUILT/AS-IS BIM .....	23
2.5.1	<i>Parametric Libraries</i> .....	25
2.5.2	<i>Automation for As-Built/As-Is BIM</i> .....	27
2.5.3	<i>Quality Control for As-Built/As-Is BIM</i> .....	31
2.6	PROCEDURAL MODELLING .....	31
2.7	COMPARISON OF EXISTING APPROACHES WITH PROPOSED PROCEDURAL HBIM SOLUTION .....	35
2.8	CONCLUSIONS OF REVIEW .....	35

## **Part II: CONCEPTUAL DESIGN AND INTIAL IMPLEMENTATION**

<b>CHAPTER THREE: PROTOTYPE I - PROCEDURAL FAÇADE .....</b>		<b>39</b>
3.1	INTRODUCTION .....	39
3.2	OVERVIEW OF PROCEDURAL BUILDING FAÇADE .....	41
3.3	ARCHITECTURAL RULES FOR PROCEDURAL FAÇADE .....	42
3.4	DESIGN AND CONCEPTUAL FRAMEWORK .....	44
3.4.1	<i>Parametric Design</i> .....	44
3.4.2	<i>Shape Grammars</i> .....	46
3.4.3	<i>Design of Parametric Shape Grammar Elements</i> .....	48
3.5	IMPLEMENTATION WITH THE GEOMETRIC DESCRIPTION LANGUAGE .....	53
3.5.1	<i>Suitability of GDL for Shape Grammar Techniques</i> .....	54
3.5.2	<i>Encoding Shape Grammar Techniques with GDL</i> .....	54
3.5.3	<i>3D Script</i> .....	56
3.5.4	<i>Master Script</i> .....	64
3.5.5	<i>Other Scripts</i> .....	65
3.5.6	<i>Non-Uniform and Irregular Geometries</i> .....	67
3.6	SUMMARY .....	68

3.7	LIMITATIONS OF PROTOTYPE.....	68
<b>CHAPTER FOUR: PROTOTYPE II – PROCEDURAL BUILDING .....</b>		<b>70</b>
4.1	INTRODUCTION .....	70
4.2	OVERVIEW OF PROTOTYPE II: PROCEDURAL BUILDING .....	70
4.3	PROTOTYPE II: RULE AND ALGORITHMIC DESIGN .....	71
4.3.1	<i>Rule One: Procedural Extrusion.....</i>	<i>72</i>
4.3.2	<i>Rule Two: Procedural Offset.....</i>	<i>77</i>
4.3.3	<i>Rule Three: Repeat .....</i>	<i>86</i>
4.3.4	<i>Rule Four: Split Rule .....</i>	<i>90</i>
4.3.5	<i>Rule Five: Replacement Rule.....</i>	<i>100</i>
4.4	PROTOTYPE II: IMPLEMENTATION OF PROCEDURAL RULES.....	101
4.5	PROTOTYPE II: SUMMARY .....	116
4.6	PROTOTYPE II: LIMITATIONS .....	117
<b>CHAPTER FIVE: PROTOTYPE III – IRREGULAR PROCEDURAL BUILDING .....</b>		<b>118</b>
5.1	INTRODUCTION .....	118
5.2	OVERVIEW OF PROTOTYPE III: IRREGULAR PROCEDURAL BUILDING .....	118
5.3	PROTOTYPE III: RULE AND ALGORITHMIC DESIGN .....	119
5.3.1	<i>Rule One: Procedural Surface Generation from Cut-Sections .....</i>	<i>119</i>
5.3.2	<i>Rule Two: Offset Rule .....</i>	<i>131</i>
5.3.3	<i>Rule Three: Split Rule .....</i>	<i>134</i>
5.3.4	<i>Rule Four: Replacement Rule.....</i>	<i>140</i>
5.4	PROTOTYPE III: IMPLEMENTATION OF PROCEDURAL RULES.....	147
5.5	PROTOTYPE III: SUMMARY .....	153
<b>CHAPTER SIX: PROCEDURAL HBIM – MAPPING TO SURVEY DATA.....</b>		<b>154</b>
6.1	INTRODUCTION .....	154
6.2	INTEGRATION OF SURVEY DATA IN A BIM ENVIRONMENT .....	154
6.2.1	<i>New Tool for Importing Orthographic Images to ArchiCAD BIM Software .....</i>	<i>156</i>

6.2.2	<i>New Tool for Importing 3D Points to ArchiCAD BIM Software</i> .....	157
6.3	MAPPING TO SURVEY DATA – PROTOTYPE I.....	159
6.4	MAPPING TO SURVEY DATA – PROTOTYPE II .....	165
6.5	MAPPING TO SURVEY DATA – PROTOTYPE III .....	167
6.6	CONCLUSION .....	172

### **Part III: EVALUATION AND TESTING**

#### **CHAPTER SEVEN: CASE STUDIES.....174**

7.1	INTRODUCTION .....	174
7.2	CASE STUDY 1 - HENRIETTA STREET DUBLIN .....	174
7.2.1	<i>Introduction and Background to Case Study</i> .....	174
7.2.2	<i>Data Collection and Pre-Processing</i> .....	174
7.2.3	<i>Generation of BIM Geometry with the Procedural Façade Prototype</i> .....	176
7.2.4	<i>Documentation and Further Applications of Data</i> .....	176
7.2.5	<i>Review of Case Study</i> .....	181
7.3	CASE STUDY 2 - THE FOUR COURTS DUBLIN .....	182
7.3.1	<i>Introduction and Background to Case Study</i> .....	182
7.3.2	<i>Previous Restoration Works</i> .....	183
7.3.3	<i>Current Restoration Work</i> .....	184
7.3.4	<i>Laser Scan Survey &amp; Pre-Processing</i> .....	185
7.3.5	<i>Generation of BIM Geometry with the Irregular Procedural Building Prototype</i> .....	188
7.3.6	<i>Documentation Results &amp; Analysis</i> .....	193
7.3.7	<i>Adopted Conservation Methods</i> .....	198
7.3.8	<i>Review of Case Study</i> .....	202
7.4	CONCLUSION .....	203

#### **CHAPTER EIGHT: VALIDATION AND TESTING .....204**

8.1	INTRODUCTION .....	204
8.2	ACCURACY TESTING .....	205

8.2.1	<i>Accuracy Tests with Case Studies</i> .....	205
8.3	END-USER SCENARIO TESTING.....	209
8.3.1	<i>Participants for End-User Test</i> .....	210
8.3.2	<i>User-Evaluation Sessions</i> .....	211
8.3.3	<i>Design of the User-Evaluation Questionnaire</i> .....	212
8.4	LEVEL OF AUTOMATION .....	214
8.4.1	<i>Automated Stages in Workflow</i> .....	214
8.4.2	<i>Test Scenario</i> .....	215
8.4.3	<i>Quantifying the Improved Efficiency</i> .....	217
8.5	CONCLUSION .....	218

## **Part IV: RESULTS AND CONCLUSIONS**

<b>CHAPTER NINE: RESEARCH FINDINGS AND ANALYSIS</b> .....		<b>221</b>
9.1	INTRODUCTION .....	221
9.2	FINDINGS OF ACCURACY TESTS.....	221
9.2.1	<i>Physical Measurement Method</i> .....	221
9.2.2	<i>Deviation Analysis Method</i> .....	224
9.3	FINDINGS OF END-USER SCENARIO TESTING.....	239
9.3.1	<i>Participant Characteristics and End-User Requirements</i> .....	239
9.3.2	<i>Evaluation of Current BIM software and Workflows</i> .....	243
9.3.3	<i>Evaluation of New Procedural HBIM Approach</i> .....	247
9.4	LEVEL OF AUTOMATION FINDINGS .....	252
9.5	DISCUSSION.....	254
9.5.1	<i>Accuracy of Procedural HBIM Techniques</i> .....	254
9.5.2	<i>Efficiency of Procedural HBIM Techniques</i> .....	255
9.5.3	<i>Usability of Procedural HBIM Techniques</i> .....	256
9.6	CONCLUSION .....	257

<b>CHAPTER TEN: CONCLUSIONS AND FUTURE WORK .....</b>	<b>259</b>
10.1 INTRODUCTION .....	259
10.2 CONCLUSIONS .....	259
10.3 SUMMARY OF CONTRIBUTIONS.....	261
10.4 LIMITATIONS AND RECOMMENDATIONS FOR FURTHER RESEARCH.....	262
10.4.1 <i>Software Prototype Limitations &amp; Recommended Solutions</i> .....	262
10.4.2 <i>Recommendations for Further Research</i> .....	264
<b>REFERENCES .....</b>	<b>266</b>
<b>APPENDICES .....</b>	<b>273</b>

## TABLE OF FIGURES

Figure 1.1: Workflow diagram for the research methodology.....	11
Figure 2.1: Sample historic data and parametric library objects as part of HBIM. ....	26
Figure 2.2: Automatically segmented point cloud showing different planar regions in different colours .....	28
Figure 3.1: New prototypes for procedural modelling with ArchiCAD BIM software..	39
Figure 3.2: Workflow for new procedural HBIM approach. ....	40
Figure 3.3: Proportions for façades and openings. ....	43
Figure 3.4: Testing classical proportions using orthographic images from surveyed classical buildings.....	43
Figure 3.5: Proportions and parameters used for design of parametric façade. ....	44
Figure 3.6: Modifying parameters from a dialogue box (left) and modifying parameters graphically (right).....	45
Figure 3.7: Initial shape <i>I</i> for parametric shape grammar design.....	48
Figure 3.8: Basic shape vocabulary elements <i>S</i> for parametric shape grammar design. ....	49
Figure 3.9: Repeated wall tile resulting from the application of rule 8 to shape <i>TW</i> .....	51
Figure 3.10: Application of shape rules specified in Table 3.2.....	52
Figure 3.11: Another application of shape rules specified in Table 3.2 to create ashlar block wall detail. ....	52
Figure 3.12: GDL code which randomly generates 3D content shown in Figure 3.13. ....	55
Figure 3.13: Automatically generated 3D content with GDL using random parameters and transformations. ....	55
Figure 3.14: GDL code for vocabulary shape <i>TW</i> .....	57
Figure 3.15: Sample script for parametric sash window stored as a vocabulary shape in a subroutine.....	58
Figure 3.16: Script for a parametric sill stored as a vocabulary shape in a subroutine. ....	58
Figure 3.17: Sample script for ashlar block wall stored as a vocabulary shape in a subroutine.....	59
Figure 3.18: Parameters as an array to allow multiple assignments for a parameter. ....	60
Figure 3.19: Sample code showing coordinate transformations embedded in a loop as part of rule eight (Table 3.2). ....	61
Figure 3.20: Shape rule 10 which adds a window object ( <i>W</i> ) to all existing window tiles ( <i>TW</i> ). ....	62



Figure 3.21: Parametric façade showing block wall detail automatically added. ....	63
Figure 3.22: Graphical editing of the façade width. Graphical editing points shown with purple markers.....	63
Figure 3.23: Sample GDL code for graphical parameter editing. ....	64
Figure 3.24: Sample GDL code showing parametric expressions used to define classical proportions. ....	65
Figure 3.25: Sample GDL code used to alter groups of parameters for simultaneously editing. ....	66
Figure 3.26: Various façade models automatically generated by altering parameters of the new procedural façade template. ....	67
Figure 4.1: Procedural Modelling Prototypes for generating building geometry. ....	70
Figure 4.2: Rule One – Extrude a building footprint or user-drawn 2D polygon to create a mass model .....	72
Figure 4.3: Inputs and outputs from procedural rule one – Procedural Extrusion .....	73
Figure 4.4: Flow diagram showing steps for rule one – Procedural Extrusion .....	74
Figure 4.5: Diagram showing solution for calculating arc origins using arc start and end points and arc angle ( $\theta_1$ ). ....	75
Figure 4.6: Rule Two - Offset a building footprint to convert mass model into walls with uniform thickness.....	78
Figure 4.7: Inputs and outputs from procedural rule two – Procedural Offset .....	79
Figure 4.8: Flow diagram showing steps for rule two – Procedural Offset .....	80
Figure 4.9: Original polygon (left), unconnected offset lines and arcs (middle) and single connected offset polygon (right) .....	81
Figure 4.10: Flow diagram showing steps for calculating the intersections of offset lines and arcs. ....	82
Figure 4.11: Intersection of two line using Equation 4.11. ....	82
Figure 4.12: Diagram showing solution for calculating the intersection of a line and arc (Table 4.3). ....	83
Figure 4.13: Diagram showing solution for calculating the intersection of two arcs (Table 4.5). ....	85
Figure 4.14: Rule Three – Repeat instances of a wall for any number of floors as set by a user. ....	87
Figure 4.15: Inputs and outputs from procedural rule three – Repeat Rule .....	87
Figure 4.16: Flow diagram showing steps for rule three – Repeat Rule.....	88

Figure 4.17: Interactive editing the number of floors by clicking and dragging a hotspot. .....	89
Figure 4.18: Flow diagram showing steps for calculating the number of floors after interactive editing. ....	89
Figure 4.19: Rule Four - Split all floors or a specific floor on a building side into any number of tiles set by a user.....	90
Figure 4.20: Inputs and outputs from procedural rule four – Split Rule.....	91
Figure 4.21: Flow diagram showing steps for rule four – Split Rule. ....	92
Figure 4.22: Diagram showing solution for calculating split line coordinates for subdividing a planar building façade (Equation 4.20).....	93
Figure 4.23: Diagram showing solution for calculating split line coordinates for subdividing a curved building face.....	94
Figure 4.24: Wall tile ( <i>TW</i> ) containing opening showing local origin and relative coordinate systems aligned to the plane of tile. ....	95
Figure 4.25: Diagram for step one in Table 4.8 to calculate distance to window points on planar tiles (lines shown in red).....	96
Figure 4.26: Diagram for step two in Table 4.8 to calculate absolute coordinates of window points on linear tiles (points shown in blue). ....	96
Figure 4.27: Diagram for step four in Table 4.8 to calculate intersection points shown in red. ....	97
Figure 4.28: Diagram for step five in Table 4.8 to calculate remaining window opening points shown with blue points. ....	97
Figure 4.29: Final window plan positions calculated after applying steps in Table 4.8.	98
Figure 4.30: Procedural rules applied to curved geometry. ....	99
Figure 4.31: Tiles referenced by their semantic position on a building. Tiles are numbered left to right on each floor. In this image, the first tile on each floor is highlighted in blue and additional tiles are highlighted in green. ....	99
Figure 4.32: Rule Five – Replacement rule to add new parametric library objects or shapes and replace existing geometry.....	100
Figure 4.33: Replacement rule to automatically add objects such as a parametric floor slab between floors. ....	101
Figure 4.34: New HBIM menu to access functions of prototype plug-in (top). A selected 2D footprint (bottom left) is used to generate a building model (bottom right). ....	104

Figure 4.35: New HBIM menu to access functions of prototype plug-in (top). A building footprint is drawn by a user (bottom left) which automatically generates a building model (bottom right). .....	104
Figure 4.36: Sample GDL code showing an executive script (located at the top of a 3D script) used to call subroutines for generating block models or wall models. ....	105
Figure 4.37: Sample GDL code showing a subroutine which generates geometry for a mass model vocabulary shape ( <i>MM</i> ) using a <i>cPRSIM</i> function.....	106
Figure 4.38: GDL code to calculate the new number of floors when a building height has been graphically edited by a user. ....	107
Figure 4.39: Example of an extensible two-dimensional array in ArchiCAD for storing splitLine_y coordinates. ....	111
Figure 4.40: Various building models automatically generated with the rules and algorithms for the procedural building prototype II. ....	112
Figure 4.41: Parametric building models automatically generated from 2D building footprints for an area of Dublin city centre. ....	113
Figure 4.42: Parametric building models automatically generated from 2D building footprints for an area of Dublin city centre. ....	114
Figure 4.43: IFC building models automatically generated from 2D building footprints for an area of Dublin city centre.....	115
Figure 5.1: Prototype III for generating irregular building geometry containing deformation. ....	118
Figure 5.2: Rule One – Procedural surface generation from any number of horizontal cut-sections.....	120
Figure 5.3: Multiple cut-sections through a point cloud used to model irregular and non-vertical wall geometry. ....	120
Figure 5.4: GDL <i>RULED</i> function which creates a shape that connects two polygons at different heights with the same number of nodes.....	121
Figure 5.5: GDL <i>RULED</i> function used to create non-vertical circular geometry between two polygon definitions. Circular geometry created by estimating a 2D arc or circle with straight line segments. ....	122
Figure 5.6: Point cloud section through a circular wall (left) and same point cloud section converted to a polygon containing sixteen arcs (right) .....	123
Figure 5.7: Inputs and outputs for rule one – Procedural Surface Generation.....	123

Figure 5.8: Flow diagram showing steps for rule one – Procedural Surface Generation .....	124
Figure 5.9: Flow diagram showing steps for the standardisation algorithm. ....	126
Figure 5.10: Diagram showing a framework of standard radial lines between four horizontal cut-sections. ....	127
Figure 5.11: Diagram showing framework of standard radial lines created for a section comprising of four arcs. ....	128
Figure 5.12: Diagram showing the solution for finding which arcs intersect with each standard radial line.....	129
Figure 5.13: Diagram showing solution for calculating the intersection points between standard radial lines and arcs in each section.....	130
Figure 5.14: Diagrams showing a low-resolution estimate of 2D arcs by straight line segments which connect standardised nodes. ....	131
Figure 5.15: Rule Two – Offset an irregular external wall surface by a uniform distance to create a higher level of detail model containing both internal and external wall surfaces. ....	132
Figure 5.16: Polygon definition for external and internal wall surfaces using a low resolution (16 line segments) to estimate the true arcs. ....	133
Figure 5.17: Rule Three – Split rule to subdivide an irregular wall surface into floors and tiles containing openings. ....	134
Figure 5.18: Flow diagram showing the steps for rule three –Split Rule. ....	136
Figure 5.19: Diagram showing the subdivision of a wall surface using split angles to calculate split line x and y coordinates. ....	138
Figure 5.20: Diagram showing solution for calculating the first two window opening points on a subdivided circular wall surface. ....	139
Figure 5.21: Diagram showing solution for calculating the remaining two window opening points on a subdivided circular wall surface.....	139
Figure 5.22: Diagram showing current (red) and new position (blue) of window opening after editing a parameter for the window opening start angle.....	140
Figure 5.23: Rule Four – Replacement Rule to add new parametric library objects or shapes and replace existing shapes. ....	140
Figure 5.24: Replacement rule used to add parametric windows (W) to existing circular wall tiles (TW). ....	141

Figure 5.25: Replacement rule used to add additional parametric objects to a generated wall or building model. Arch-top niches, rectangular niches and windows are added using this replacement rule. ....	142
Figure 5.26: Primitive shapes used create arch-top niche objects which are cut from a wall using Boolean Operations.....	143
Figure 5.27: Polygon outline used to define a rectangular niche object which is cut from a wall surface using Boolean operations.....	143
Figure 5.28: Diagram showing solution for calculating an arc centre point using three points on the arc.....	144
Figure 5.29: Diagram showing the solution for calculating coordinates for window, arch-top niche and rectangular niche objects on an irregular circular wall surface. ....	145
Figure 5.30: Library objects for a column shaft automatically generated from survey data (cut-sections) with procedural rule one (top) or alternatively ideal columns created from architectural data, added to existing geometry with a replacement rule (bottom).....	146
Figure 5.31: New HBIM menu to access functions of prototype plug-in (top). Nine polygon sections are selected (bottom left) which are used to generate the non-vertical wall containing deformation (bottom right). ....	148
Figure 5.32: Sample GDL code from the 3D script where a <i>RULED</i> command is used to generate the geometry for an external wall surface from any number of sections. ....	149
Figure 5.33: Various models automatically generated from cut-sections with the rules and algorithms for the Irregular Procedural Building prototype III. ....	151
Figure 5.34: Various surfaces automatically generated from cut-sections with the surface generation rule for the Irregular Procedural Building prototype III.....	152
Figure 6.1: Geo-referenced orthographic images imported into ArchiCAD BIM software in their correct 3D position. ....	156
Figure 6.2: Sample GDL code for new tool to import geo-referenced orthographic imagery. ....	157
Figure 6.3: 3D points represented with GDL using very small line segments.....	158
Figure 6.4: Cut-sections and segmented point clouds imported to ArchiCAD BIM software in correct 3D position. ....	159

Figure 6.5: Menu command for accessing new tools to accurately import survey data into ArchiCAD BIM software.....	159
Figure 6.6: Orthographic image showing a plan view of a building façade and entrance created from laser scan survey data. ....	160
Figure 6.7: Initial configuration of a procedurally generated façade. ....	161
Figure 6.8: Measurements taken from survey data (left) and entered as parameters into dialogue box for procedural façade model.....	161
Figure 6.9: Graphical parameter editing to procedurally generate the structure of a façade. Parameter for the number of storeys edited (top) and the number of horizontal tiles edited (bottom). ....	163
Figure 6.10: Graphical parameter editing to modify door position by moving a hotspot in the 2D window. ....	163
Figure 6.11: Orthographic image overlaid with procedural façade model for graphical editing of façade elements.....	164
Figure 6.12: Graphical editing of objects on the procedural façade model. Editing of individual window (top left), editing of all window heights on a floor simultaneously (top right) and editing façade corner (bottom).....	164
Figure 6.13: Graphically editing window opening which automatically updates linked elements such as ashlar block wall detail. ....	165
Figure 6.14: Graphical editing the “number of storeys” (left) and the building footprint (right) by clicking and dragging hotspots. ....	166
Figure 6.15: Graphically editing an individual window opening in the 3D window by clicking and dragging a window hotspot to a new position. ....	167
Figure 6.16: Cut-sections from a point cloud used to refine procedurally generated geometry (3D window).....	169
Figure 6.17: Cut-sections from a point cloud used for refining procedurally generated objects in a 2D (plan) window. ....	169
Figure 6.18: Simultaneous editing of all procedurally generated objects in a model based on angles from an average arc origin. An angle based editable hotspot is used to rotate all objects around the irregular wall structure (2D plan window). ....	170
Figure 6.19: Simultaneous editing of all procedurally generated objects in a model based on angles from an average arc origin. An angle based editable hotspot is used to rotate all objects around the irregular wall structure (3D window).....	170

Figure 6.20: A hotspot located in the middle of an object used to move a single object at once. This alters an individual objects start and end angles to move the object around a circular wall structure. ....	170
Figure 6.21: An angle type hotspot at either side of an object is used edit the width of an individual object. The width is calculated based on a new start or end angle of the object.....	171
Figure 6.22: A hotspot located at the bottom of an object is used to change the formation level of an object. ....	171
Figure 6.23: A hotspot located at the top of an object is used to change the height of an individual object. ....	171
Figure 7.1: Trimble GS200 laser scan used to record data for generating an as-is BIM of Henrietta Street.....	175
Figure 7.2: Henrietta Street, Dublin (left) and point cloud of street coloured by intensity (right). ....	175
Figure 7.3: Building information model created using the procedural façade prototype and HBIM library objects for Henrietta Street, Dublin. ....	177
Figure 7.4: Lists of objects, components and bill of quantities generated from the HBIM of Henrietta Street.....	178
Figure 7.5: Automated 2D and 3D documentation produced from HBIM. ....	179
Figure 7.6: Automated 2D documentation from BIM model.....	180
Figure 7.7: Energy analysis performed on HBIM data for Henrietta Street using EcoDesigner STAR extension for ArchiCAD BIM software. ....	180
Figure 7.8: War damage to the Four Courts (left) and 18 <sup>th</sup> -century section drawing of the Four Courts by architect James Gandon (right).....	182
Figure 7.9: Damage to the Four Courts after the bombardment by the Free State Army, during the Civil War on 30 June 1922 (Cashman, 1922b). ....	183
Figure 7.10: Part of a damaged Corinthian capital that broke away and fell onto the roof below (top left). Damaged Corinthian capital (top right) and rusted steel ring encircling the dome (bottom). ....	184
Figure 7.11: Exterior view of the Four Courts Dublin where the dome is undergoing major restoration (Byrne, 2015). ....	185
Figure 7.12: Point cloud of The Four Courts showing points coloured by their intensity values. ....	186

Figure 7.13: Complete registered point cloud of the Four Courts and surrounding buildings coloured by intensity values.....	187
Figure 7.14: Point cloud showing detailed scans of capitals and the upper dome after scaffolding was put in place. ....	187
Figure 7.15: Cut-sections taken through the point cloud for the Four Courts dome and drum. Polygon sections shown are in red. ....	188
Figure 7.16: Irregular non-vertical circular walls automatically generated from polygon cut-sections.....	189
Figure 7.17: Non-vertical wall generated with 24 tiles containing alternating niche and window objects (left). Internal dome surface and floor surface automatically generated as mesh surfaces from segmented point clouds (right).....	190
Figure 7.18: True condition of column shafts automatically generated from cut-sections. ....	190
Figure 7.19: True condition of beam encircling dome automatically generated from cut-sections.....	191
Figure 7.20: Historic Building Information Model (HBIM) components for the Four Courts automatically generated using the new HBIM procedural rules. ....	191
Figure 7.21: Triangulated mesh models of a Corinthian capital from the Four Courts which was captured using photogrammetry. A photo textured mesh is shown on the left and an un-textured mesh model is shown on the right.....	192
Figure 7.22: HBIM of the Four Courts with imported triangulated mesh model for a Corinthian capital representing its true condition. ....	193
Figure 7.23: 2D plan vector drawing automatically generated from the 3D model for the Four Courts, Dublin. ....	194
Figure 7.24: 2D plan vector drawing showing a column that is leaning (left). Documentation shows vertical walls (left) and areas of the wall and beam that are leaning (right).....	195
Figure 7.25: Section through the Four Courts drum and dome automatically generated from the 3D model.....	195
Figure 7.26: Section through the Four Courts drum automatically generated from the 3D model. ....	196
Figure 7.27: Section showing the true condition of walls which contains deformation (vertical wall lines shown by dashed red lines).....	196



Figure 7.28: Vector elevation drawing automatically generated from the 3D model for the Four Courts, Dublin. ....	197
Figure 7.29: Variation between an ideal vertical drum and the actual drum which contains deformation. Areas coloured green have the highest variation while areas coloured blue have a low variation. ....	198
Figure 7.30: Reflected internal plaster dome of the Four Courts Rotunda viewed from below. Photo containing distortion (left) and true to scale orthographic image generated from point cloud (right). ....	199
Figure 7.31: Location of steel trusses supporting the internal plaster dome is marked on the orthographic image (left). Cut-sections of same steel trusses from the point cloud are shown on the right. ....	200
Figure 7.32: Orthographic image of a Corinthian capital from the Four Courts Dome (right). Vector documentation manually created from orthographic image is shown on the left. ....	200
Figure 7.33: Current renovation work to the Four Courts dome, drum and capitals (Byrne, 2015). ....	201
Figure 7.34: Scaffolding put in place in the top dome for resurfacing the internal concrete surface (Byrne, 2015). ....	201
Figure 7.35: Current renovation work to the Four Courts dome, drum and capitals (Byrne, 2015). ....	202
Figure 8.1: Total Station used to capture reference measurements (left), measurements viewed in AutoCAD software (middle) and coordinates viewed in excel (right). ....	206
Figure 8.2: Measuring coordinates from a procedurally generated virtual façade model within the ArchiCAD software. ....	206
Figure 9.1: Laser scan point cloud used as a reference for accuracy test (left) and procedurally generated BIM model converted to OBJ format (right). ....	224
Figure 9.2: Graphical and statistical results from an absolute deviation analysis between drum wall point cloud and procedurally generated model. ....	226
Figure 9.3: Graphical and statistical results from a signed deviation analysis between drum wall point cloud and procedurally generated model. ....	227
Figure 9.4: Photo (left) and procedurally generated model component for a beam encircling the drum wall. ....	228

Figure 9.5: Graphical and statistical results from an absolute deviation analysis between a reference point cloud and procedurally generated model of the beam encircling the drum wall.....	229
Figure 9.6: Graphical and statistical results of a signed deviation analysis between a reference point cloud and procedurally generated model of the beam encircling the drum wall.....	230
Figure 9.7: Graphical and statistical results of an absolute deviation analysis between a reference point cloud and procedurally generated model of the columns surrounding the drum wall. ....	232
Figure 9.8: Graphical and statistical results of a signed deviation analysis between a reference point cloud and procedurally generated model of the columns surrounding the drum wall. ....	233
Figure 9.9: Graphical and statistical results of an absolute deviation analysis between a reference point cloud and all procedurally generated components. ....	234
Figure 9.10: Graphical and statistical results of a signed deviation analysis between a reference point cloud and all procedurally generated components. ....	235
Figure 9.11: Combined point cloud and procedurally generated BIM model used for accuracy testing. ....	236
Figure 9.12: Cut-sections through point cloud (coloured yellow and orange) and generated BIM geometry (coloured blue) used for a visual accuracy assessment. ....	237
Figure 9.13: Cut-sections through point cloud (coloured yellow and orange) and generated BIM geometry (coloured blue) used for a visual accuracy assessment. ....	238
Figure 9.14: Results of end-user scenario testing - “Participants Area of Work”.....	240
Figure 9.15: Results of end-user scenario testing - “Participants Organisation” and “Participants in Academia and Industry”.....	241
Figure 9.16: Results of end-user scenario testing – “Experience with Modelling Existing Building/Conservation Projects”.....	242
Figure 9.17: Results of end-user scenario testing – “Main Modelling Software Used” and “Preferred Source Data for Modelling Existing Buildings”.....	242
Figure 9.18: Results of end-user scenario testing – “End-User Accuracy Requirements”. ....	243

Figure 9.19: Results of end-user scenario testing – “Adequacy of’ Current Software Tools for Modelling Existing Buildings”.	244
Figure 9.20: Results of end-user scenario testing – “Efficiency of’ Current Software Tools for Modelling Existing Buildings”.	244
Figure 9.21: Results of end-user scenario testing – “Greatest Challenge when Modelling Existing Buildings”.	245
Figure 9.22: Results of end-user scenario testing – “Automation Used” and “Need for Automation”.	246
Figure 9.23: Results of end-user scenario testing – “Usefulness of Procedural Modelling for Modelling Existing Buildings”.	246
Figure 9.24: Results of end-user scenario testing – “Usefulness of New Procedural HBIM Prototypes for Modelling Existing Buildings”.	247
Figure 9.25: Results of end-user scenario testing – “Most Useful Feature of New Procedural HBIM Prototypes”.	248
Figure 9.26: Results of end-user scenario testing – “Efficiency of New Procedural HBIM Prototypes”.	248
Figure 9.27: Results of end-user scenario testing – “Usability of New Procedural HBIM Prototypes”.	249
Figure 9.28: Results of end-user scenario testing – “Accuracy of New Procedural HBIM Prototypes”.	250
Figure 9.29: Results of end-user scenario testing – “Suitability of HBIM Deliverables for Conservation Projects”.	250
Figure 9.30: Results of end-user scenario testing – “Most Suitable Deliverables for Conservation Projects”.	251

## TABLE OF TABLES

Table 2.1: Terrestrial laser scanning methods (Barber and Mills, 2007).....	17
Table 2.2: Comparison of existing automated modelling approaches with proposed Procedural HBIM approach. ....	35
Table 3.1: Key parameters designed for modifying the façade template.....	46
Table 3.2: Shape rules $R$ for parametric shape grammar design. ....	50
Table 4.1: Steps for calculating arc origins using arc start and end coordinates and arc angle ( $\theta_1$ ) (Figure 4.5).....	75
Table 4.2: Equations used in rule one, Procedural Extrusion. ....	76
Table 4.3: Steps for calculating the intersection of a line and arc (Figure 4.12).....	83
Table 4.4: Equations used to calculate the intersection points between a line and arc. .	84
Table 4.5: Steps for calculating the intersection of two arcs (Figure 4.13). ....	85
Table 4.6: Equations used to calculate the intersection points between two arcs .....	86
Table 4.7: Equations for subdividing a curved building face.....	94
Table 4.8: Steps for calculating window openings on curved building faces. ....	95
Table 4.9: C++ Functions for implementing rule one – Procedural Extrusion .....	102
Table 4.10: Steps for function one “ <i>Do_editLibraryObject</i> ” shown in Table 4.9.....	102
Table 4.11: Steps for function two “ <i>Do_editLibraryObject_UserInput</i> ” shown in Table 4.9. ....	103
Table 4.12: Example of a two-dimensional array for storing multiple values of a window width parameter.....	109
Table 4.13: Example of new method for storing extensible parameter values in arrays for implementing procedural rule four-Split Rule.....	110
Table 5.1: Steps for calculating the intersection points between standard radial lines and arcs in each section (Figure 5.13).....	128
Table 5.2: Steps to calculate the x and y coordinates for window openings on an irregular circular wall.....	137
Table 5.3: C++ Functions for implementing rule one – Procedural Surface Generation from Cut-Sections.....	147
Table 5.4: Steps for function one “ <i>Do_editLibraryObjectSection</i> ” shown in Table 5.3. .....	148

Table 9.1: Deviations between total station measurements and a procedurally generated building façade model for number 3 Henrietta Street. ....	222
Table 9.2: Deviations between total station measurements and laser scan point cloud measurements. ....	222
Table 9.3: Deviations between laser scan point cloud and a procedurally generated building façade model for number 3 Henrietta Street. ....	223
Table 9.4: Additional feed on procedural HBIM prototypes. ....	251
Table 9.5: Results from Level of Automation Testing. ....	253
Table 9.6: Results from Level of Automation Testing - Time saving and percentage difference in accuracy. ....	253

## **LIST OF APPENDICES**

Appendix A: Information Sheet for Participants of End-User Scenario Test .....	273
Appendix B: Consent Form for End-User Scenario Test .....	275
Appendix C: End-User Scenario Questionnaire 1 .....	276
Appendix D: End-User Scenario Questionnaire 2 .....	282
Appendix E: End-User Scenario Questionnaire 3 .....	288
Appendix F: Results of physical measurement accuracy test showing deviations between total station measurements and a procedurally generated building façade model for number 3 Henrietta Street. ....	295
Appendix G: Additional check for physical measurement accuracy test to verify deviations between total station measurements and laser scan point cloud measurements. ....	296
Appendix H: Additional check for physical measurement accuracy test to verify deviations between laser scan point cloud and a procedurally generated building façade model for number 3 Henrietta Street. ....	297

## **GLOSSARY**

3D – Three Dimension

API – Application Programming Interface

As-Built – Documentation of a building/structure after it is built

As-Is – Up-to-date documentation of a building/structure at the time of survey

AEC – Architecture, Engineering and Construction

BIM – Building Information Modelling

BREP – Boundary Representation

CAD – Computer Aided Design

CGA – Computer Generated Architecture

CityGML – City Geographic Markup Language

COBie – Construction Operations Building Information Exchange

CRP – Close Range Photogrammetry

CRS – Coordinate Reference System

CSG – Constructive Solid Geometry

ESRI - Environmental Systems Research Institute

FM – Facility Management

GDL – Geometric Description Language

GML – Geographic Markup Language

GML – Generative Modelling Language

GNSS – Global Navigation Satellite System

GPS – Global Positioning System

GIS – Geographic Information System

HBIM – Historic Building Information Modelling

IFC – Industry Foundation Classes

KML – Keyhole Markup Language

LiDAR – Light Detection and Ranging

LoA – Level of Accuracy

LoD – Level of Detail

LoI – Level of Information

MVD – Model View Definition

Point Cloud – A set of data points in a particular coordinate system.

TLS – Terrestrial Laser Scanning

# **Part I**

# **INTRODUCTION**

**Part I Summary:**

Part I of this thesis contains two chapters which provide an introduction to the research. These two chapters include a description of the research problem, aims and objectives, scope of research, contributions of research, research methodology and a review of existing literature relating to the research.

## **Chapter One: Introduction**

### **1.1 Background, Motivation and Context**

*“Architectural heritage is our legacy from the past, what we live with today, and what we pass on to future generations. Our architectural heritage is an irreplaceable source of life and inspiration”* (UNESCO, 2011). It is crucial to record and document our architectural heritage for many reasons, one of the most important is to maintain and preserve history to create a record for current and future generations. Understanding the history and current condition of buildings and landscapes allows for an informed decision on conservation (Eppich et al., 2007). The International Committee for Heritage Documentation (CIPA, 2010) believe that a monument cannot be restored and protected until it has been fully measured and documented.

The first step to preservation of our architectural heritage is documentation. Without understanding and knowing everything possible about a historic building, site or object it is impossible to develop a sound rational and effective plan for preserving it. A major problem with this is how to best construct and disseminate knowledge about heritage sites that can lead to the most effective methods of preservation.

There is an increased demand for documenting existing buildings with digital information enhanced 3D models. Along with a building’s geometry, information enhanced models can also contain information about a buildings semantics, topology, relationships between components and attributes. The main motivation for documenting our built heritage with smart information enhanced models is the wide variety of applications that the information-rich models can be used for. This includes applications for documentation and management of buildings along with great capabilities for energy, structural and economic analysis of buildings.



In order to create an accurate representation of existing buildings, accurate and up-to-date survey data is required. There are many surveying and photogrammetric techniques available for recording the information needed to create accurate 3D models of existing buildings. Laser scanning technology has become very popular for collecting information for generating accurate models of existing buildings. This is due to the speed at which it can capture data, the level-of-detail and accuracy in the resulting point cloud information. Other traditional survey technologies such as total stations cannot record the same level of detail at the speed possible with laser scanning. Photogrammetric techniques are also very well suited to capturing data for generating 3D models of existing buildings. Photogrammetric techniques use images taken at different viewpoints to record the 3D geometry of a building or object. Photogrammetric techniques can produce similar results to laser scanning such as point clouds, mesh models and orthographic imagery.

The generation of accurate 3D models of existing buildings can be divided into three main stages. This includes data acquisition, pre-processing of survey data and 3D modelling. The third stage of modelling is the longest stage in this process. This final modelling stage is a reverse engineering process where geometric components are created and mapped to the survey data to create the 3D model.

Developments in CAD modelling have led to the introduction of a new concept called Building Information Modelling (BIM). Unlike previous CAD modelling, BIM incorporates object-oriented, parametric and feature-based modelling concepts combined with the addition of a dynamic 3D database for storing information relating to buildings. Due to its many benefits, BIM has received a lot of attention in both industry and academia. As BIM was designed for modelling and representing new and modern buildings the focus of this attention to date has mainly been on the use of BIM in the

planning, design and construction stages of a building (Volk et al., 2014). Recently however, there has been a shift in BIM related research from early life cycle stages to maintenance, refurbishment and management of buildings throughout their complete lifecycle (Volk et al., 2014). The benefits of BIM make it a very suitable solution for modelling and managing information relating to a building after the construction stages and throughout a building's lifecycle. A BIM for an existing or historical building can be used as a documentation tool for conservation work, retrofitting, renovations or as a tool for performing building analysis.

The concepts of an as-built or as-is BIM are relatively new concepts that are being used to describe the recording of existing buildings with BIM. As-built drawings or record drawings are typically submitted by a contractor after completion of a project. These drawings should reflect any deviations from the original design made during the construction process. Due to the benefits of BIM, as-built or record drawings are now being replaced by an as-built BIM. After the construction phase, 3D laser scanning or other survey data capture methods are used to collect data needed to generate an accurate and up-to-date as-built BIM.

For many conservation or renovation projects, an accurate or up-to-date as-built BIM or drawings are not available. In these situations, an as-is BIM can be created from survey data. An as-is BIM reflects the true condition of a building at the time of survey. Another name given to the process of generating BIM geometry from laser scan data is 'Scan to BIM'. However, as stated by Thomson and Boehm (2015) the name 'Scan to BIM' is wrongly formed as the end result is not a BIM process, but a 3D parametric model that aids the process at its current level of development.

## 1.2 Research Problem

Although BIM greatly improves the process of modelling and recording information relating to a building, there are still a number of problems with the use BIM for modelling existing buildings. Three main problems are identified:

1. The current process for reconstructing BIM geometry from point clouds is very much a manual one. This manual process is recognised as being time-consuming, tedious, subjective and costly (Thomson and Boehm, 2015, Volk et al., 2014, Tang et al., 2010).
2. Current BIM software lack specific tools for modelling existing buildings:
  - a. Current BIM software is not suitable for accurately representing non-uniform and irregular geometries often occurring in existing buildings such as walls out of plumb.
  - b. Current BIM software lacks pre-defined components suitable for existing buildings. Most native and 3<sup>rd</sup> party BIM libraries are focused only on modern buildings. As a result, modelling existing and historic buildings often require many bespoke components to be created from scratch which can be a very time-consuming process.
3. Due to the lack of tools for modelling existing buildings, more advanced workflows and software are required to accurately model buildings as they actually exist. This requires costly training and high levels of skill to be competent in these workflows and processes.

The lack of automation and tools for accurately modelling existing buildings with BIM results in a time-consuming, complex and costly process. To overcome this, more

automated solutions need to be developed for BIM which are capable of accurately modelling existing and historical buildings from survey data.

### **1.3 Research Question**

This research will assess if an automated modelling approach could be developed to improve the current problems with reconstructing BIM geometry from point clouds. Automating the scan to BIM process is a primary focus for a lot of researchers. Jung et al. (2014), Xiong et al. (2013) and Zhang and Zakhor (2014) have shown promising results for automatic object recognition and feature extraction from point clouds. Although progress has been made in this area, results are currently limited to automatic extraction of basic elements such as planes and openings. Automatic extraction of complex architectural elements of existing and historical buildings is still in its infancy.

Another automated approach which has not been used for AEC and BIM applications is procedural modelling. A procedural modelling approach uses a sequence of generation instructions, rules or algorithms that can be repeated with varying characteristics to automatically generate varying 3D geometries (Kelly and McCabe, 2006). Procedural modelling has many advantages such as automatic generation of geometry, great flexibility for variation and object hierarchy.

Based on the current research problems and the potential of a procedural modelling approach, the main question this research is attempting to answer is:

***“Can procedural modelling techniques provide tools for more accurate, automated and easier generation of BIM geometry from point clouds?”***

A more accurate, automated and easier solution for converting point clouds to BIM geometry would facilitate more efficient generation of high quality heritage

documentation. This would facilitate better decision making for building rehabilitation and management of existing and historical buildings throughout a building lifecycle.

#### **1.4 Aim and Objectives of Research**

The aim of this research is to address the following research hypothesis; *“Procedural modelling is a suitable solution for more accurate, automated and easier generation of BIM geometry from point clouds”*. The objectives of the research are:

- To develop workflows and tools for combining captured survey data in a BIM environment.
- To design a grammar of parametric shapes and objects that can be used to reconstruct building geometry from point clouds.
- To design new procedural rules and algorithms for reconstructing façade and building geometry from point clouds.
- To design efficient methods of interactive editing to alter procedurally generated geometry.
- To implement the designed grammar of shapes, procedural rules and methods for interactive geometry editing as a prototype plug-in for existing BIM software.
- To quantitatively evaluate the new procedural modelling tools with relevant case studies.
- To design and implement end-user scenario testing for a qualitative evaluation of the suitability of the new procedural modelling tools.
- To evaluate results of testing and redesign plug-in and process.

## 1.5 Scope of Research

As the modelling of existing architecture can involve many research areas, it is necessary to define the scope of the research that is to follow.

- The modelling of existing architecture from point clouds can be divided into three main stages. This includes data acquisition, pre-processing of survey data and 3D modelling. The focus of this research is on the final modelling stage. Extensive research has been previously carried out on data capture and pre-processing techniques (Bryan et al., 2009, Boehler and Heinz, 1999, Grussenmeyer and Hanke, 2010).
- The process of BIM involves managing many aspects of a building such as materials, characteristics, cost, scheduling, behavioural and energy properties. The focus of this research is on generating BIM geometry which also contains semantic and attribute data. This reconstructed BIM geometry can then be used to aid further BIM processes where additional building properties are added.
- Depending on the application of a building information model, different levels of detail are required. The level of geometric and semantic detail in a model need to be decided based on the intended use. Construction and heritage applications tend to require precise geometric reconstructions while lower accuracy models containing geometric generalisations may be more suitable for applications such as simulation and operational management. The focus of this research is on generating models for construction and heritage applications so high levels of detail and accuracy are required.
- There are many different approaches towards automation for modelling existing buildings. The focus of this research is on a procedural modelling approach.

Other automated approaches such as point cloud feature extraction are not explored in the scope of this research.

- This research concentrates on accurately reconstructing historical architecture for further management and analysis with BIM. The focus of the library of architectural elements for building reconstructions is confined to the classical period in the 18<sup>th</sup> and early 19<sup>th</sup> centuries in Ireland. However, the procedural rules developed are applicable to many architectural styles and could be used in the future with library objects for other architectural styles.

## **1.6 Contributions**

The new contributions to knowledge from this research include:

- An extensive review of existing research into the reconstruction of BIM geometry from point clouds.
- Novel tools for accurately importing survey datasets into a BIM environment.
- New parametric shapes and objects for reconstructing BIM geometry from point clouds.
- Novel procedural modelling rules and algorithms developed as plug-ins to existing BIM software. This includes procedural rules for:
  - Automatically generating standard vertical wall objects from floor plans.
  - Automatically generating non-vertical wall objects from multiple cut-sections.
  - Automatically splitting linear and curved façades into floors and tiles.
  - Automatically generating and positioning architectural objects on façade/building tiles.

- Tools for interactive editing of computer-generated geometry. This includes efficient tools for smart editing of objects simultaneously in groups or individually.

The procedural modelling rules and algorithms developed from this research enable the automatic generation of geometry containing detailed and complex objects. Previous procedural modelling implementations, which were not developed for BIM, are inefficient at generating detailed and complex geometry. As a result, previous procedural modelling implementations were limited to applications for visualisation for industries such as film and gaming. The development of procedural modelling rules which are capable of generating complex objects enable the tools to be used for new applications in the AEC and heritage sectors.

## **1.7 Research Methodology**

The hypothesis of this research is that procedural modelling is a suitable solution for more accurate, automated and easier generation of BIM geometry from point clouds. In order to gather information to test this hypothesis, the methodology involved designing and developing new procedural modelling tools as plug-ins to existing BIM software. Testing was then carried out to test the hypothesis and to develop software plug-ins. The adopted methodology for testing the research hypothesis involved both quantitative and qualitative research methods. Data from real cases studies was used to quantitatively evaluate the accuracy capabilities of the new procedural modelling approach. A qualitative evaluation of the new procedural modelling approach was also achieved from end-user testing. The final part of the methodology involved analysing the results after testing. This allowed the suitability of a procedural modelling approach to be evaluated. The prototypes could then be redesigned based on the results and feedback (Figure 1.1).



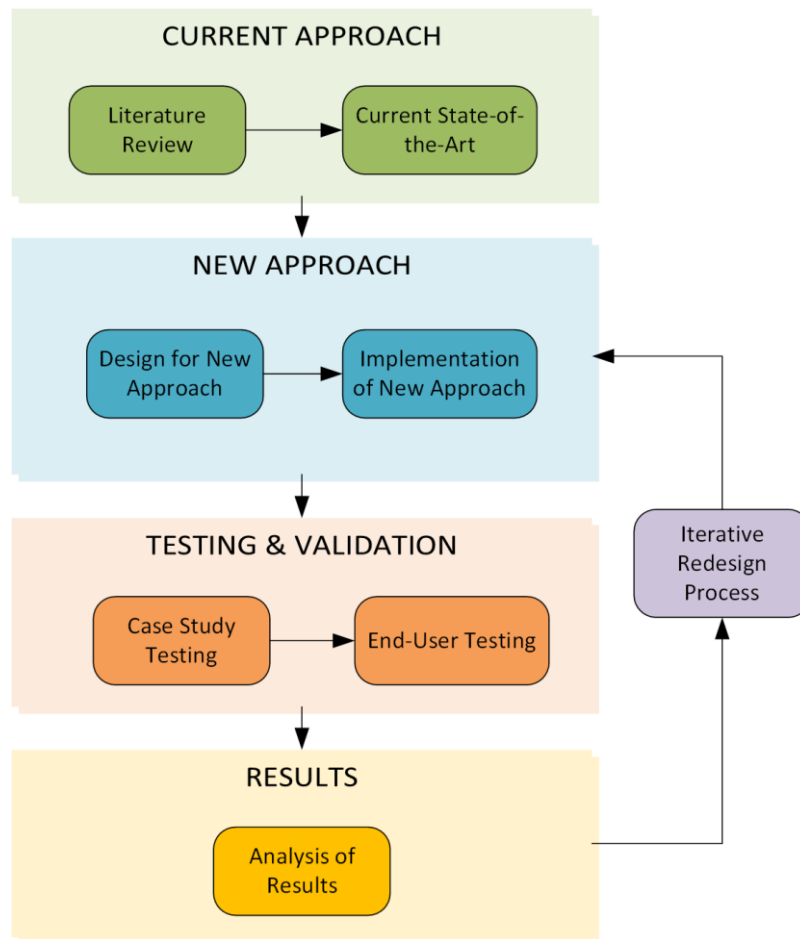


Figure 1.1: Workflow diagram for the research methodology.

## 1.8 Publications

Throughout this research, several publications have been made on different parts of the work. Below is the list of peer-reviewed publications achieved:

### 2012

Dore, C. and Murphy, M. 'Integration of HBIM and 3D GIS for Digital Heritage Modelling', *Digital Documentation 2012*, Edinburgh, Scotland, 22 -23 October 2012.

Dore, C. and Murphy, M. 'Integration of Historic Building Information Modeling and 3D GIS for Recording and Managing Cultural Heritage Sites'. *18<sup>th</sup> International Conference on Virtual Systems and Multimedia: "Virtual Systems in the Information Society"*, Milan, Italy, 2-5 September 2012: IEEE, 369-376.

## 2013

Dore, C. and Murphy, M. 'Laser Scan to BIM - A New Approach for Generating As-Built Building Information Models from Point Cloud Data'. *CITA BIM Gathering*, Dublin, Ireland, 14th-15th November 2013.

Dore, C. and Murphy, M. 'Semi-Automatic Modelling of Building Façades with Shape Grammars using Historic Building Information Modelling'. *3D-ARCH 2013 – 3D Virtual Reconstruction and Visualization of Complex Architectures*, Trento, Italy, 25 -26 February 2013: ISPRS Archives, 57-64.

Dore, C. and Murphy, M. 'Semi-Automatic Techniques for As-Built BIM Facade Modelling of Historic Buildings'. *Digital Heritage International Congress (DigitalHeritage)*, 2013, Oct. 28 2013-Nov. 1 2013, 473-480.

## 2014

Dore, C. and Murphy, M. (2014a) '*Semi-Automatic Generation of As-Built BIM Facade Geometry from Laser and Image Data*', *Journal of Information Technology in Construction*, 19, pp. 20-46.

Dore, C. and Murphy, M. 'Semi-Automatic Techniques for Generating BIM Façade Models of Historic Buildings', *Virtual Cultural Heritage in Ireland 2014*, Dublin, 27th-28th February 2014.

## 2015

Dore, C. and Murphy, M. (2015) 'Historic Building Information Modelling (HBIM)', in Brusaporci, S. (ed.) *Emerging Digital Tools for Architectural Surveying, Modeling, and Representation*: IGI Global.

Dore, C., Murphy, M., McCarthy, S., Brechin, F., Casidy, C. and Dirix, E. (2015) 'Structural Simulations and Conservation Analysis -Historic Building Information Model (HBIM)', *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XL-5/W4, pp. 351-357.

## 1.9 Format of Document

This dissertation contains ten chapters which are organised into four parts as described below.

### Part I: INTRODUCTION

Part I of this thesis contains two chapters which provide an introduction to the research.

These two chapters include the current chapter which describes the research problem,

aims and objectives, scope of research, contributions of research and research methodology. The second chapter in Part I, contains a review of existing literature relating to the research.

## **Part II: CONCEPTUAL DESIGN AND INTIAL IMPLEMENTATION**

Part II of this thesis contains four chapters which describes the conceptual design and initial implementation of three new prototypes for procedural modelling with Historic Building Information Modelling (HBIM).

## **Part III: EVALUATION AND TESTING**

Part III of this thesis contains two chapters which describe the methodology for testing and validating the new concepts of procedural HBIM. In the first chapter of Part III, two case studies are undertaken to fully implement and validate the procedural HBIM concepts using real world applications. The second chapter of Part III describes three tests undertaken for further validation of the procedural HBIM concepts and prototypes.

## **Part IV: RESULTS AND CONCLUSIONS**

The final part of this thesis, Part IV, contains two chapters which describe the findings, analysis and conclusions of the research.

## **Chapter Two: Current State of the Art**

### **2.1 Introduction**

The research reviewed in this chapter shows the current state-of-the-art for generating BIM geometry from point clouds and provides a context for the research that is to follow. This review covers the following range of topics:

1. Heritage Documentation Standards
2. Data Collection and Pre-Processing Techniques
3. 3D Modelling Concepts
4. As-Built/As-Is BIM
  - a. Parametric Libraries
  - b. Automation for As-Built/As-Is BIM
  - c. Quality Control for As-Built/As-Is BIM
5. Procedural Modelling

### **2.2 Heritage Documentation Standards**

*“Heritage is the full range of our inherited traditions, monuments, objects, and culture. Most important, it is the range of contemporary activities, meanings, and behaviours that we draw from them” (UMass Amherst Center, 2017).*

The International Council on Monuments and Sites (ICOMOS), define cultural heritage as “an expression of the ways of living developed by a community and passed on from generation to generation, including customs, practices, places, objects, artistic expressions and values” (ICOMOS, 2012). Cultural heritage can include tangible elements of our built environment, natural environment and artefacts such books, documents, objects and pictures. Cultural heritage can also include intangible forms

such as values, traditions, oral history, traditional skills, technologies, religious ceremonies, performing arts and storytelling (Teijgeler, 2017). The focus of this research is on improving methods for documenting elements of our architectural heritage.

Heritage documentation is the systematic collection and archiving of both tangible and intangible elements of historic structures and environments. The purpose of documentation is to supply accurate information that will enable correct conservation, monitoring and maintenance for the survival of an artefact (Eppich et al., 2007, Bryan et al., 2009). Standards have been developed for heritage documentation at regional, national and international levels.

At an international level recording standards have been established by the International Council on Monuments and Sites (ICOMOS), which is a non-governmental organisation of professionals, committed to the conservation of the world's historic monuments and sites (ICOMOS, 2012). ICOMOS operates through national committees and scientific committees. One such committee is CIPA, the International Committee for Architectural Photogrammetry. CIPA was established in collaboration with the International Society of Photogrammetry and Remote Sensing (ISPRS). The main aim of CIPA is to improve methods for surveying cultural monuments and sites. The work of CIPA has been instrumental in developing new automated methods of digital recording and storage in addition to the standards required for accuracy of surveying and documentation of built heritage (CIPA, 2010). “*RecorDIM*” is a CIPA initiative in collaboration with other international heritage conservation organisations to improve and develop standards for the documentation of architectural heritage (Eppich et al., 2007).

The United Nations Educational, Scientific and Cultural Organisation (UNESCO) also provide a directive for recording cultural heritage. This states that

*“documentation must factually represent the building or site; the tools used should be means for monitoring the condition through time; and that the record should be easily interpreted” (UNESCO, 2011).*

By achieving this objective heritage documentation can provide crucial information that is needed to assist with preservation and restoration of our cultural heritage.

Examples of national standards for collecting and archiving information relating to historic structures are detailed in national guidelines such as the Historic American Building Surveys (HABS) and the English Heritage Metric Survey Practice (Balachowski, 2005, Bryan et al., 2009). Recently, the authors of the London Charter proposed a set of guidelines for the use of virtual reality for representation and presentation for cultural heritage (Beacham et al., 2009). The charter seeks to outline the requirements for computer visualisation and proposes that the process and product be valid and transparent.

At the regional level, researchers at the Carleton Immersive Media Studio have carried out research on the development of standards for its use in heritage applications. A lot of effort is being made to establish standards for BIM within the AEC industry; however, this is even more challenging for heritage applications due to the complex and irregular nature of existing buildings. In an application of BIM for the documentation and management of the West Block of Canada’s Parliament Hill, Fai and Rafeiro (2014) have established an appropriate level of detail (LoD) for as-built/as-is BIM projects. For this project, Fai & Rafeiro (2014) suggest three levels of detail for effective long-term use which are based on the AEC Canada guidelines.

## 2.3 Data Collection and Pre-processing Techniques

There are many surveying techniques available for acquiring data needed to generate accurate as-built and as-is Building Information Models. This includes the use of terrestrial laser scanning (TLS), photogrammetry and other traditional survey equipment such as total stations and GPS/GNSS equipment. A review of these techniques is outlined in this section.

### 2.3.1 Terrestrial Laser Scanning

Terrestrial laser scanning (TLS) is one of the most efficient methods of collecting data for accurate as-built/as-is modelling of buildings (Allen et al., 2003, Boehler and Heinz, 1999, Grussenmeyer and Hanke, 2010, Bernardini and Rushmeier, 2002). TLS can automatically record millions of three-dimensional points on an object in near real time. TLS measures distances and angles from the sensor to an object being scanned with millimetre to centimetre accuracies possible. TLS operates on three different principals which are; triangulation, time of flight and phase comparison. All three types of laser scanners produce a 3D point cloud of the object. However, the range and accuracy capable from each method vary (Table 2.1). TLS provides an accurate, efficient and easy to use solution for acquiring 3D data required for as-built/as-is modelling of buildings. The main disadvantage of TLS for this application is the high cost of this technology.

Table 2.1: Terrestrial laser scanning methods (Barber and Mills, 2007)

<b>Terrestrial Laser Scanning Methods</b>			
<i><b>TLS Method</b></i>	<i><b>Range</b></i>	<i><b>Accuracy</b></i>	<i><b>Use</b></i>
Triangulation	<3m	<1mm	Small Objects
Time of Flight	2 – 200m	<5mm	Large Objects/Scenes
Phase Comparison	2 – 100m	5 – 10mm	Large Scenes

### **2.3.2 Photogrammetry**

Photogrammetry is the art and science of determining accurate measurements and three-dimensional data from photographs (Matthews, 2008). Photogrammetric techniques use images taken at different viewpoints to record the 3D geometry of a building or object. Photogrammetric techniques are becoming very popular for recording existing buildings, especially for cultural heritage applications. Low-cost digital cameras, powerful computer processing and the greater availability of commercial and open source photogrammetric software are driving many new applications for this technology (Beraldin, 2004). One of the main advantages of photogrammetry over laser scanning is the addition of high-quality imagery and colour information to the resulting data. The principles of photogrammetry are similar for both aerial and close range (ground based) photogrammetry. The main principles of photogrammetry are based on triangulation where lines of sight (rays) from two different camera locations are joined to a common point on the object. The intersection of these rays determines the three-dimensional location of the point. Using this technique with two images is known as stereo photogrammetry. When more than two images are used a bundle adjustment is used to simultaneously calculate all the unknown parameters. Although these techniques can be carried out with low-cost digital cameras, the entire process required can be cumbersome with high processing times. The outputs from photogrammetric surveys are similar to the products obtained from laser scanning and include orthographic images, point clouds, triangulated surface models and also textured surface models.

### **2.3.3 Other Survey Techniques**

Traditional survey equipment such as Total Stations and GPS/GNSS equipment can provide very accurate measurements but at a much slower rate than laser scanning and photogrammetric surveys. Unlike laser scanning and photogrammetric data which can be collected in near real time, traditional survey equipment such as Total Stations and



GPS/GNSS require each individual point to be manually recorded. These slower methods of data collection would not be economical for large as-built/as-is projects especially in the cultural heritage field where millions of points are often required to accurately record a complex building or structure. Although Total Station and GPS/GNSS methods may not always be appropriate as the main method of data collection, they are often still required in addition to laser scanning and photogrammetry to record accurate control points needed to process laser and image data.

#### **2.3.4 Pre-Processing Laser and Image Data**

Raw data acquired from terrestrial laser scanning (TLS) and close range photogrammetry (CRP) both require a number of pre-processing steps in order to generate products that can be used to create 3D CAD and BIM models. One of the main differences between TLS and CRP is that TLS automatically captures 3D point clouds directly while CRP requires pre-processing to generate 3D point clouds from images captured on site.

Even though TLS captures 3D point clouds directly a number of pre-processing steps are still required. Because most objects cannot be scanned from one single scan position, individual scans must be accurately combined and referenced together. This stage is called “registration”. This requires common targets or points to be identified in different scans. Developments in laser scan processing software have led to increased levels of automation for this step which includes automatic and semi-automatic target detection in separate scans. Research is also being carried out on full automatic techniques for registering laser scan point clouds (Kima et al., 2016, Sun et al., 2016). Other pre-processing of TLS data include segmenting point clouds and filtering out unwanted data. Automatic triangulation of 3D points can also be carried out to create a mesh surface model from the 3D point cloud. This 3D mesh surface model can then be

used to generate orthographic images by combining the 3D surface model with 2D images. 3D mesh models can also be textured using referenced image data. 2D cut sections and 3D vectors can also be generated from the 3D point cloud or 3D surface model.

Pre-processing for close range photogrammetry (CRP) varies depending on the number images. The main methods of processing include stereo processing and multi-convergent processing (bundle adjustment). Common processing stages required for both methods include selecting common feature points between images, calculating camera positions, orientations, distortions and reconstructing 3D information by intersecting feature point locations (Klein et al., 2012). Developments in computer vision and image matching algorithms have allowed for many of these steps to be carried out automatically. An example of an automated approach can be seen in work by Barazzetti et al. (2010). Although developments have been made towards full automatic pre-processing for close range photogrammetry, the accuracy and quality of the results of most automatic techniques cannot yet match the results of manual or semi-automatic procedures (Gruen, 2012, Lynch et al., 2016). Using CRP for high accuracy as-built/as-is modelling, requires human interaction in the pre-processing of image data.

## **2.4 3D Modelling Concepts**

Advances in survey technology now allow for very fast and efficient data collection methods. Pre-processing of survey data is also becoming increasingly automated (Kima et al., 2016, Sun et al., 2016). 3D modelling from remote sensor data is still, however, a manual and long process with much demand for new automated solutions.

Two important advancements in 3D CAD modelling introduced in the 1970s and 1980s were the concepts Constructive Solid Geometry (CSG) and Boundary Representation

(BREP) (Gomes and Teixeira 1991). CSG uses solid primitive shapes to represent objects. This approach is more powerful than previous wireframe approaches as solid objects can be used to calculate various physical properties such as volume, density, weight and mass. CSG also allows solid primitive shapes to be combined using Boolean operations such as union, subtract and intersect to create more complicated shapes. Alternatively, Boundary Representation (BREP) represents objects by describing their faces, edges, vertices and topology. BREP also includes operations such as extrude, sweep and revolve which can be used to create 3D shapes from 2D outlines. Many CAD software platforms incorporate both BREP and CSG modelling concepts to provide greater flexibility for modelling complex objects. 3D shapes represented with CSG and BREP methods exist only as graphic entities and do not have intelligence (Ibrahim and Krawczyk, 2004).

The next evolutionary stage in 3D modelling is the introduction of parametric and feature-based modelling which introduced a certain amount of intelligence into model elements (Shah and Mantyla, 1995). Feature-based modelling is an object orientated approach where in addition to geometry, objects contain information about the objects role (e.g. door, wall, window etc.) and how an object relates to other objects. Feature-based modelling allows operations such as creating holes, fillets and chamfers to be associated with objects. This could include a window automatically cutting a hole when placed in a wall or intersecting walls connecting and joining correctly. Feature-based modelling enables objects to interact with other objects correctly and automatically in a spatial environment (Leeuwen and Wagter, 1997).

Parametric modelling differs from standard 3D CAD modelling as objects such as primitive shapes are associated with parameters or variables that can instantly change the geometry or other properties of that object. Simple parameters of an object may

include the length, width, height, or radius of the object. Other more complex parametric objects may have parameters that can change the entire structure or geometry of an object depending on different conditions. Parameters of an object can also control the location of an object within a larger model. Parametric library objects (such as doors or windows) allow objects to be reused multiple times in a model or in many different models with varying parameters. This approach is very efficient for modelling elements that are repeated but may contain geometric variation between different instances (Baik et al., 2014).

The more recent development of the concept Building Information Modelling (BIM) incorporates the main developments in 3D modelling including parametric and feature-based modelling combined with a dynamic 3D database for storing information relating to buildings. The addition of a dynamic relational database for building elements (similar to a Geographic Information System) enables many new applications for managing and analysing building elements. BIM enables building elements to be documented with smart parametric reusable objects that contain rich information about the objects use, semantics, topology, relationships with other objects and further information stored as attributes. BIM can be defined as the assembling of parametric objects which represent building components within a virtual environment and which are used to create or represent an entire building (Murphy, 2012). Objects are described according to parameters some of which are user-defined and others, which relate to its position in a 3D environment relative to other shape objects. The visualisation of objects is achieved through viewing 2D and 3D features, plans, sections, elevations and 3D views. BIM can be used to automatically create cut-sections, elevations, details and schedules in addition to orthographic projections and 3D models (wireframe or textured and animated). All of these views are linked to the 3D model and automatically update in real time, so if a change is made in one view, all other views are also updated. This

enables efficient generation of detailed documentation required in the AEC/FM and heritage industries.

The open standard data format, Industry Foundation Classes (IFC) was developed by BuildingSMART International to facilitate interoperability between different BIM authoring software (BuildingSMART International, 2013). IFC is a vendor-neutral, open and standardised data model for the representation of complex building models. The IFC data model is based on STEP, the ISO standard for the exchange of product model data. STEP includes the specification of the data modelling language EXPRESS, which is employed for defining the IFC schema (BuildingSMART International, 2013).

Another important BIM standard used for facility management is the Construction Operations Building Information Exchange (COBie). COBie is an information exchange specification for life-cycle capture and delivery of information required by facility managers (BuildingSMART Alliance, 2011). COBie is a model view definition (MVD), which is a subset of IFC for information relating to facility management.

## **2.5 As-Built/As-Is BIM**

The benefits of BIM make it a very suitable solution for modelling and managing information relating to existing buildings. A BIM for an existing or historical building can be used as a documentation and management tool for conservation work, retrofitting, renovations and building analysis. The concepts of an as-built or as-is BIM are being used to describe the recording of existing buildings with BIM. An ‘as-built’ representation is the recording of a building after construction and an ‘as-is’ representation is the recording of a building at a particular moment in time. Both of these processes involves three main stages; data acquisition, pre-processing of survey data and a modelling stage.

Volk et al. (2014), Hichri et al. (2013), Tang et al. (2010), Liu et al. (2017) and Barbosa et al. (2016) provide excellent and comprehensive overviews on the current state-of-the-art in the area of as-built/as-is BIM. The fundamental problems outlined in these reviews are the “high modelling/conversion effort” required for creating semantic BIMs from unstructured survey data, the difficulties in accurately representing the variety of complex and irregular objects occurring in existing buildings and the lack of standards for the representation of objects and information in existing buildings.

A lot of developments for as-built/as-is BIM are emerging from the adoption of BIM from cultural heritage communities. Many authors have shown the benefits of using BIM for cultural heritage preservation (Fai et al., 2011, Oreni et al., 2014, Wu et al., 2013, Boeykens et al., 2012, Pauwels et al., 2008, Quattrini et al., 2015, Barazzetti et al., 2015). Pauwels et al. (2008) describe an approach called Architectural Information Modelling which uses BIM to document geometric data along with appended historical information such as photographs, scanned documents or research material. Fai et al. (2011) adopt a similar approach which links heritage information to a BIM but also includes documentation related to tangible and intangible heritage. Boeykens et al. (2012) use BIM software to create a reconstruction of the Vinohrady Synagogue in Prague which was demolished in 1951. The authors of this paper note that the BIM software used was almost completely focussed on contemporary buildings and that more specific tools are needed for historical reconstructions. Another development emerging from the cultural heritage community is a plug-in to Autodesk Revit called “*GreenSpider*” (Garagnani and Manferdini, 2013). This plug-in improves the current process for importing unstructured datasets into BIM software by translating key points from the point cloud into native reference snaps in Revit.

### **2.5.1 Parametric Libraries**

Most BIM software packages have extensive libraries of pre-defined parametric objects that are used to create 3D building information models. This facilitates efficient modelling as 3D geometry does not have to be created from scratch. Instead existing information enhanced library objects can be used to model the main building elements such as walls, doors, windows, columns, beams, slabs, roofs etc. Parameters of these library objects are edited to match the required dimensions and settings of a project. These library objects are then combined to create a complete model. A major problem for as-built/as-is BIM is the lack of pre-defined parametric objects suitable for existing and historical buildings. Most native and 3<sup>rd</sup> party BIM libraries are focused only on modern buildings. As a result, modelling existing and historic buildings often require many bespoke components to be created from scratch which can be a very time-consuming process.

This limitation of BIM for existing buildings has motivated research in the development of new parametric libraries that would be suitable for existing and historic buildings. Various projects (Baik et al., 2014, Chevrier et al., 2010, Fai and Rafeiro, 2014, Murphy et al., 2013, De Luca, 2012) have shown that the development of new reusable parametric library objects supports high levels of detail (LoD) while decreasing the time of modelling. The parametric objects created by Fai & Rafeiro (2014) were created for the Autodesk Revit BIM software and contain very useful objects representing gothic style architectural windows. The library created by Baik et al. (2014), also for Autodesk Revit BIM software, contains parametric objects for heritage projects in the Al-Balad district of Jeddah City. An extensive library of parametric objects was created by Chevrier et al. (2010). However, these objects are not suitable for BIM and were instead created with the MEL scripting language for Autodesk Maya software.

Work from Murphy et al. (2013) has led to the development of a new library of parametric objects for BIM software called Historic Building Information Modelling (HBIM) (Figure 2.1). These new library objects are designed for modelling classical architectural elements found in many existing buildings. The parametric architectural objects are designed from historic manuscripts and architectural pattern books and are implemented using an embedded programming language within the ArchiCAD BIM software called the Geometric Description Language (GDL). Also included with this library of objects is a system for mapping objects to survey data.

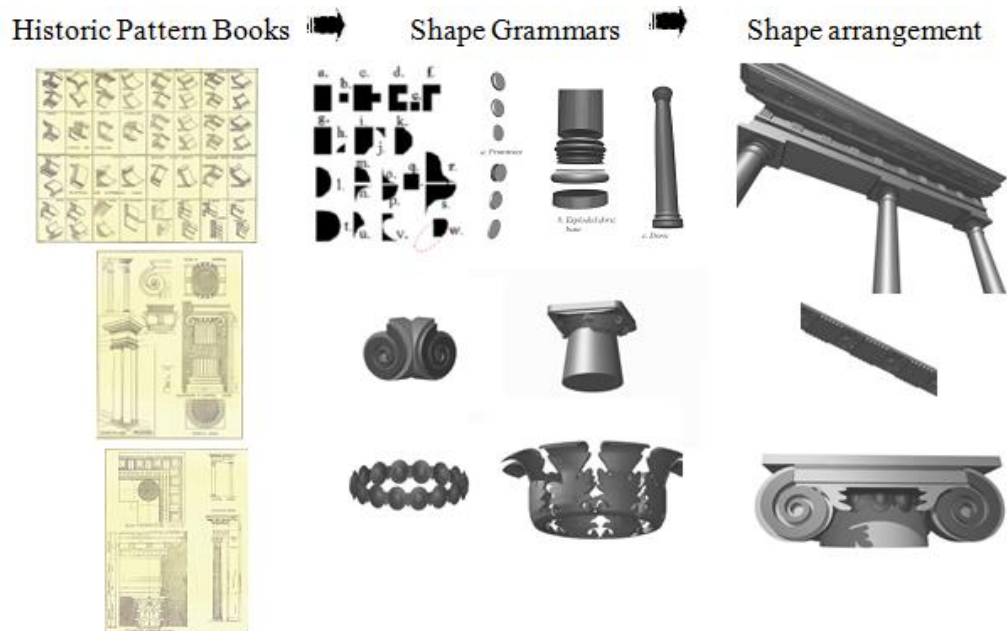


Figure 2.1: Sample historic data and parametric library objects as part of HBIM.

The research presented in the following chapters is a continuation of this work originally developed by Murphy et al. (2013) (Murphy, 2012) and Fai et al. (2011). The new research presented later further develops the HBIM library objects by using procedural modelling techniques to automatically combine and create 3D BIM content from a combination of the HBIM parametric library objects and new vocabulary shapes.



### **2.5.2 Automation for As-Built/As-Is BIM**

Automated modelling of existing buildings from scan data is very desirable commercially to reduce time and therefore costs of as-built/as-is BIM projects (Thomson and Boehm, 2015). One area where progress is being made towards automated as-built/as-is BIM is in automated object recognition and feature extraction from point clouds (Jung et al., 2014, Xiong et al., 2013, Zhang and Zakhor, 2014). Object recognition is the problem of automatically labelling data points or segments of an image with a named object or object class (Tang et al., 2010), for example automatically detecting windows or doors from an image or point cloud. Once objects are automatically recognised from a dataset, primitives or planes can be automatically fitted to the recognised elements in the dataset. Much of the research in this area, however, is focused on indoor applications (Jung et al., 2014, Previtali et al., 2014, Zhang and Zakhor, 2014, Thomson and Boehm, 2015) and is currently restricted to automatic extraction of basic elements such as planes and openings. Automatic extraction of complex architectural elements that occur in existing and historical buildings is still in its infancy.

Pu & Vosselman (2009) is one example of object recognition applied to building applications. In this work a knowledge-based approach for object recognition is adopted which aims to automatically reconstruct building facades from terrestrial laser scan data. Point clouds obtained from terrestrial laser scanning are first automatically segmented into planar surfaces which are then automatically classified as semantic features such as walls, doors, windows and roofs using generic knowledge of building facades. This includes knowledge about a features size, position, orientation and topology. Next, an outline polygon is generated for each detected feature using least squares fitting, convex hull fitting or concave polygon fitting (Figure 2.2). Finally building knowledge is again used to make assumptions for missing or occluded areas. Results of this automatic

object recognition approach include a polygon model for detected building features (Figure 2.2).

Much of the existing work for automated object recognition and feature detection show promising results but do not automate the complete process from point cloud to BIM (Hong et al., 2015, Jung et al., 2014, Xiong et al., 2013, Zhang and Zakhor, 2014, Wang et al., 2015). The results from these automatic approaches include surface models, planes, 3D vectors or a subset of the original point cloud, all of which still need to be converted into structured information enhanced and parametric BIM components which at present needs to be carried out manually (Volk et al., 2014, Thomson and Boehm, 2015). The results from Hong et al. (2015) are regularised wireframe models which are automatically detected from point clouds for indoor building applications. These simplified wireframe models are then used for geometric modelling with BIM software instead of the original point cloud. Two case studies testing this approach showed that the efficiency of the as-built/as-is BIM creation process was improved by 15.4% and 15.0% using the extracted wireframe models instead of the complete point cloud (Hong et al., 2015).



Figure 2.2: Automatically segmented point cloud showing different planar regions in different colours (left). Results of reconstructed model with detected features outlined in black (right) (Pu & Vosselman 2009).

Thomson and Boehm (2015) is one of the few examples of automated object detection and feature extraction from point clouds resulting in object-based IFC models, suitable for BIM. This approach utilises the open source Point Cloud Library (PCL) which provides a number of data handling and processing algorithms such as a RANSAC (RANdom SAMple Consensus) algorithm for automatic plane detection. The eXtensible Building Information Modelling (xBIM) toolkit, which is capable of reading and writing IFC compliant files, is used to create IFC content from detected features. While this work shows excellent progress for the automatic reconstruction of BIM geometry from point clouds, the work currently only deals with planar walls and is limited in terms of accuracy and reliability.

In conjunction with these developments in academic research, commercial software is also emerging in recent years which attempts to automate and improve existing manual workflows for the generation of BIM geometry from point clouds. This includes plug-ins to Autodesk Revit software, Trimble SketchUp software and standalone software such as Pointfuse from Arithmetica. Currently, none of these commercial software solutions are capable of fully automating the as-built/as-is BIM process but do offer various semi-automatic solutions and tools which improve the efficiency of certain stages of the process. ClearEdge3D offer a standalone software platform called EdgeWise Building in addition to a plug-in to Autodesk Revit BIM software. This software classifies a point cloud into surfaces that share coplanar points. The operator then picks floor and ceiling planes to constrain the search for extracting wall features. Once wall planes are detected the Revit plug-in creates the parametric object-based wall geometry. Another plug-in has been developed for Autodesk Revit software by IMAGINiT called Scan to BIM. This plug-in provides detection and fitting algorithms for planes and cylinders to create walls, pipes and column objects. With this approach, a

user picks three points to define a wall plane and a region growing algorithm detects the extents of the wall. The user then selects a tolerance and the type of parametric wall element to be used in the model (Thomson and Boehm, 2015).

Kubit, now owned by Faro also provide a plug-in to Autodesk Revit that detects planes for creating wall objects along with providing tools that aid the manual process of tracing points in a point cloud. Pointfuse from Arithmetica provides standalone software which can automatically detect planes and edges from a point cloud and export these in standard CAD formats. Trimble also provides an extension for their SketchUp software that can detect planes with a semi-automatic process.

These developments emerging from academic research and in commercial software platforms show progress towards automating the as-built/as-is BIM process. However, these automatic methods to date are limited to extracting simple planar features. Very little progress has been made at automatically reconstructing complex and non-planar geometry that often occurs in existing and historical buildings. Attempts at the automatic recognition of objects and feature detection are focused on indoor scenes and tend to work well only in simple and uncluttered environments. Other more complicated environments containing clutter and occlusion can result in less accurate and reliable results. Automatic object recognition and detection approaches can result in errors such as objects being incorrectly segmented and classified, objects not being classified or detected, incorrect object fitting and incorrect assumptions for occluded or missing data. Case studies from Pu & Vosselman (2009) and Thomson and Boehm (2015) show discrepancies in certain areas of ten centimetres or more between the automatically detected features and the scan data. Unfortunately, this is not currently at the level of accuracy (LoA) and reliability required for most documentation projects in the AEC and

heritage fields. Further work is needed to achieve higher accuracy results required for such applications.

### **2.5.3 Quality Control for As-Built/As-Is BIM**

For many applications of as-built/as-is BIM, it is crucial that the model accurately represents the true condition of a building. This is particularly important when BIM is used to produce documentation for conservation work, restorations, retrofitting or performing different types of building analysis. The workflows for creating as-built/as-is models also involves several steps that can be subjective in nature when carried out manually so errors can easily be introduced. Whether an as-built/as-is model is created manually or using automatic techniques it is very important that quality assurance (QA) is carried out to ensure the accuracy of the final model. Anil et al., 2011 and Anil et al., 2013 propose a new approach for QA of as-built/as-is BIM that analyses patterns of the geometric deviation between the model and the point cloud data. This research demonstrates that it is possible to identify the source, magnitude, and nature of errors by analysing the deviation patterns. This proposed deviation analysis involves computing the deviations by finding correspondences between a point cloud and model and then computing the distances between correspondences. Next, the deviations are visualised with colour-coded maps of the deviations. Finally, deviation maps are inspected and potential errors are identified and verified. In comparison to other QA approaches of using a sample of physical measurements or ground truths the deviation analysis method provides full coverage and can provide deterministic guarantees that a model completely represents the underlying data with a given accuracy specification.

## **2.6 Procedural Modelling**

Another automated approach to 3D modelling is procedural modelling. Procedural modelling is an automated approach to generating 3D content based on a sequence of

generation instructions, rules or algorithms that can be repeated with varying characteristics (Kelly and McCabe, 2006). Procedural modelling has traditionally been used in applications such as film and gaming where content can be randomly generated based on rules and algorithms. The use of shape grammars in procedural modelling has gained a lot of interest and is now being used to generate content for architectural modelling (Muller et al., 2006, Dylla et al., 2010, Hohmann et al., 2009, Becker and Haala, 2009, Becker et al., 2015). Shape grammars originally introduced by Stiny and Gips (1972) are derived from formal grammars and consist of a set of basic vocabulary shapes (terminals and non-terminals) and a set of production rules to transform these shapes to create 3D content. A shape grammar called CGA Shape (Muller et al. 2006) has been developed for the commercial software CityEngine from ESRI. This shape grammar is designed for procedural modelling of buildings and cities. CityEngine software provides users with tools to create 3D content from scripts using this shape grammar. With this software, it is possible to procedurally generate buildings from 2D footprints and other GIS datasets for modelling existing buildings and cities. While these models contain information about semantics and can be automatically generated, they lack the detail that would be required for applications in the AEC/FM and heritage communities.

Most procedural modelling applications require users to code rules in a grammar to create a model. This text-based approach restricts users with little computer science background. Work by Lipp et al. (2008) has concentrated on creating an interactive visual editing tool for shape grammars to create rules from scratch without the need for text file editing. This makes automated approaches for generating 3D content much more accessible and does not require advanced users to create scripts to code rules.

Another approach (Müller et al., 2007) uses the CGA shape grammar for automatic modelling of existing building facades from a single rectified image. This method automatically detects a façade structure using mutual information and symmetry detection to divide a façade into floors and tiles. Further tile refinement is automatically carried out using edge detection to split tiles into smaller regions using a subdivision concept from procedural modelling which creates a hierarchy of elements. This is used to automatically detect window positions, ledges and window sills. 3D objects from a library of architectural elements are then matched to the subdivided façade to add windows and other architectural elements. Depth for different sections of the façade is added manually and the computed façade can be exported as shape grammar rules in the CGA Shape Grammar. This method shows how procedural modelling techniques can be applied to existing buildings. This method works well for urban environments where facades contain a lot of repetition and symmetry can be easily detected. However less repetitive facades with a lot of architectural detail may be problematic for this automatic method.

Work by Hohmann et al. (2009) use shape grammars to automatically model building facades for automatic 3D city reconstruction. This project called “*CityFit*” aims to reconstruct 80% of the buildings in the city of Graz automatically. Their workflow uses roadside photographs and LiDAR point clouds as input data. Image based feature detection is carried out to detect and segment windows, arches and other decorative elements. Depth maps derived from the point cloud are also used to provide depth information. The results from these segmentations are matched against a set of shape grammar templates obtained by façade analysis and classification. The use of shape grammar templates in the modelling stage incorporates architectural knowledge to automate this stage. The shape grammars and shape grammar templates used are based

on the concepts of the CGA shape grammar but have been implemented using the Generative Modelling Language (GML).

A procedural modelling approach with shape grammars has many advantages such as automatic generation, great flexibility for variation, object hierarchy, scalable geometric representation and data handling of large models. However, a disadvantage of these methods for AEC and heritage applications is that they are inefficient at generating smaller complex geometric detail. Muller et al. (2006), state that manual methods are used instead of procedural modelling techniques to create detailed elements such as roof bricks, capitals and window grills. Müller et al. (2007) state that the generated models are primarily for visualisation and the aim is to automatically create a geometric model that *“looks like a plausible interpretation of the input image”*. For AEC and heritage applications, a more precise and accurate model is required. Another shortcoming of existing software for procedural modelling is an inability to automate the production of engineering drawings, which is critical for most applications in the AEC and heritage sectors.

This focus of this research is for modelling and documenting existing and historical buildings. Procedural modelling, however, can also be useful during the design and construction stages of a project. One of the benefits of procedural modelling is its flexibility for generating different building variations, arrangements and geometries. This would greatly facilitate exploring and analysing the implications of different building designs before a building is constructed. During the construction stage of a project, the original design may need to be altered due to unforeseen circumstances. In this scenario, a procedural BIM solution would also be of benefit to analyse the implications of design variations without adding major delays to a project schedule.



## 2.7 Comparison of Existing Approaches with Proposed Procedural HBIM Solution

Table 2.2 below compares features of existing automated approaches outlined in Section 2.5.2 and Section 2.6 with the proposed procedural HBIM solution. The existing automated approaches include plug-ins for scan-to-BIM modelling and existing procedural modelling applications. As seen from Table 2.2, most of the existing plug-ins and application do not contain libraries of detailed parametric objects that can be procedurally combined to generate complex and irregular building geometries.

Table 2.2: Comparison of existing automated modelling approaches with proposed Procedural HBIM approach.

	Library of Parametric Objects for Historical Buildings	Procedural Generation of Building Arrangements	Procedural Generation of Complex Architecture	Automatic Generation of Irregular Architecture	Interactive Editing of Generated Architecture	Semi-Automatic Point Cloud Feature Extraction	Full Automatic Point Cloud Feature Extraction	IFC/BIM Output
Pointfuse (Arithmetica)	✗	✗	✗	✗	✗	✓	✗	✓
EdgeWise (ClearEdge3D)	✗	✗	✗	✗	✓	✓	✗	✓
Scan to BIM (IMAGINiT)	✗	✗	✗	✗	✓	✓	✗	✓
PointSense (Faro)	✗	✗	✗	✗	✓	✓	✗	✓
CityEngine (ESRI)	✗	✓	✗	✗	✓	✗	✗	✗
Scan Explorer (Trimble)	✗	✗	✗	✗	✗	✓	✗	✗
HBIM	✓	✗	✗	✗	✓	✗	✗	✓
Procedural HBIM	✓	✓	✓	✓	✓	✗	✗	✓

## 2.8 Conclusions of Review

In this chapter, a critical review of existing work was carried out on the main topics involved in this research project. After carrying out an extensive review of existing literature a number of observations were made in relation to the current state-of-the-art and limitations of different approaches for recording and modelling existing buildings and environments:

1. Data collection and pre-processing techniques are becoming increasingly automated to allow for near real-time data capture and fast processing of this data for later modelling applications.
2. Current BIM software is almost completely focused on new buildings and has very limited tools and pre-defined libraries for modelling existing and historic buildings.
3. The development of reusable parametric library objects for existing and historic buildings supports modelling with high levels of detail while decreasing the modelling time. Mapping these parametric objects to survey data, however, is still a time-consuming task that requires further research.
4. Promising developments have been made towards automatic object recognition and feature extraction from point clouds for as-built/as-is BIM. However, results are currently limited to simple and planar features. Further work is required for automatic accurate and reliable reconstruction of complex geometries from point cloud data.
5. Procedural modelling can provide an automated solution for generating 3D geometries but lacks the detail and accuracy required for most as-built/as-is applications in AEC and heritage fields. The main applications of procedural modelling include visualisation for film or gaming. Existing procedural modelling implementations also do not automate the production of engineering drawings for construction or conservation documentation.

A new solution is presented in the following chapters to further develop the existing methods reviewed in this chapter. The main gap identified from this review is the lack of automation and specialised tools for modelling existing buildings with BIM software. Other approaches such as procedural modelling have shown promising developments

but are currently not accurate or detailed enough for AEC and heritage applications. The research presented in the following chapters includes the development a new solution which incorporates the advantages of automated procedural modelling combined with the advantages of detailed parametric modelling using parametric libraries. This new approach differs from existing work (Thaller et al., 2011, Chevrier et al., 2010, Müller et al., 2007) as it implements procedural and parametric modelling techniques in BIM software that can generate both accurate and detailed models with a semi-automatic process.

# **Part II**

## **CONCEPTUAL DESIGN AND INITIAL IMPLEMENTATION**

**Part II Summary:**

Part II of this thesis contains four chapters which describes the conceptual design and initial implementation of three new prototypes for procedural modelling with Historic Building Information Modelling (HBIM).

## Chapter Three: Prototype I - Procedural Façade

### 3.1 Introduction

The hypothesis of this research is that procedural modelling is a suitable solution for more accurate, automated and easier generation of BIM geometry from point clouds. In order to test this hypothesis the methodology first involved designing and implementing new procedural modelling tools for generating BIM geometry from point clouds. These new procedural modelling tools have been designed and implemented as three separate prototype plug-ins to Graphisoft's ArchiCAD BIM software (Figure 3.1).

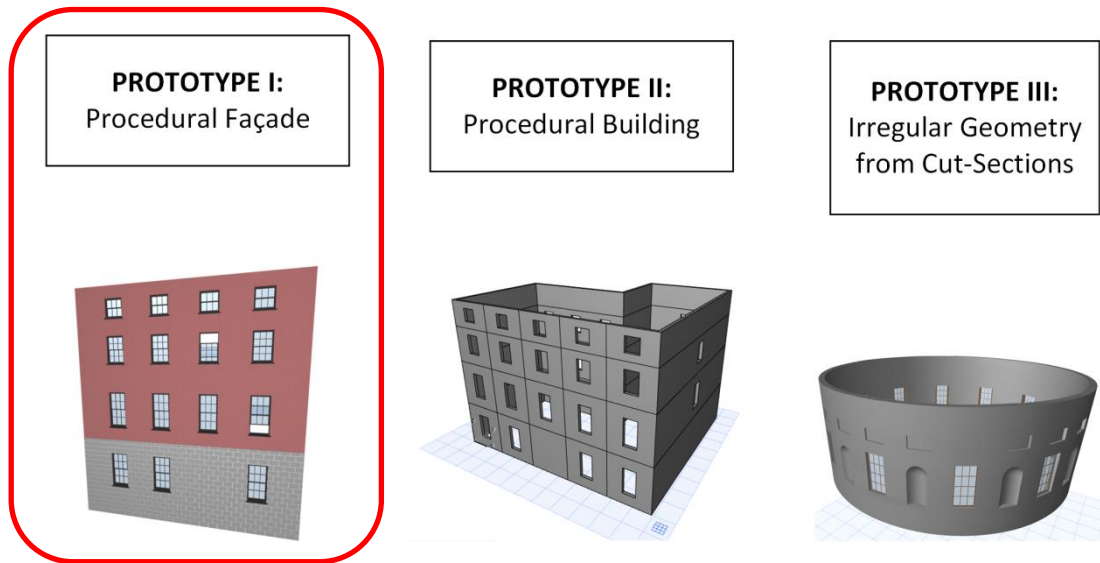


Figure 3.1: New prototypes for procedural modelling with ArchiCAD BIM software.

These three new prototypes build on previous work from Murphy et al. (2013), which developed a new process for documenting historic structures called Historic Building Information Modelling (HBIM). HBIM is a novel prototype library of parametric objects, based on historic architectural data, in addition to a mapping system for plotting the library objects to remotely sensed data (Figure 2.1). This previous work greatly

improved the process of documenting historic structures; however, it still required library objects to be manually combined for larger arrangements and mapped to survey data. This current research further develops the HBIM process by using procedural modelling techniques to automatically combine and create BIM content from HBIM library objects and survey data. The new procedural modelling rules also provide tools for modelling deformation and irregular objects. Figure 3.2 shows the workflow for the new procedural HBIM approach.

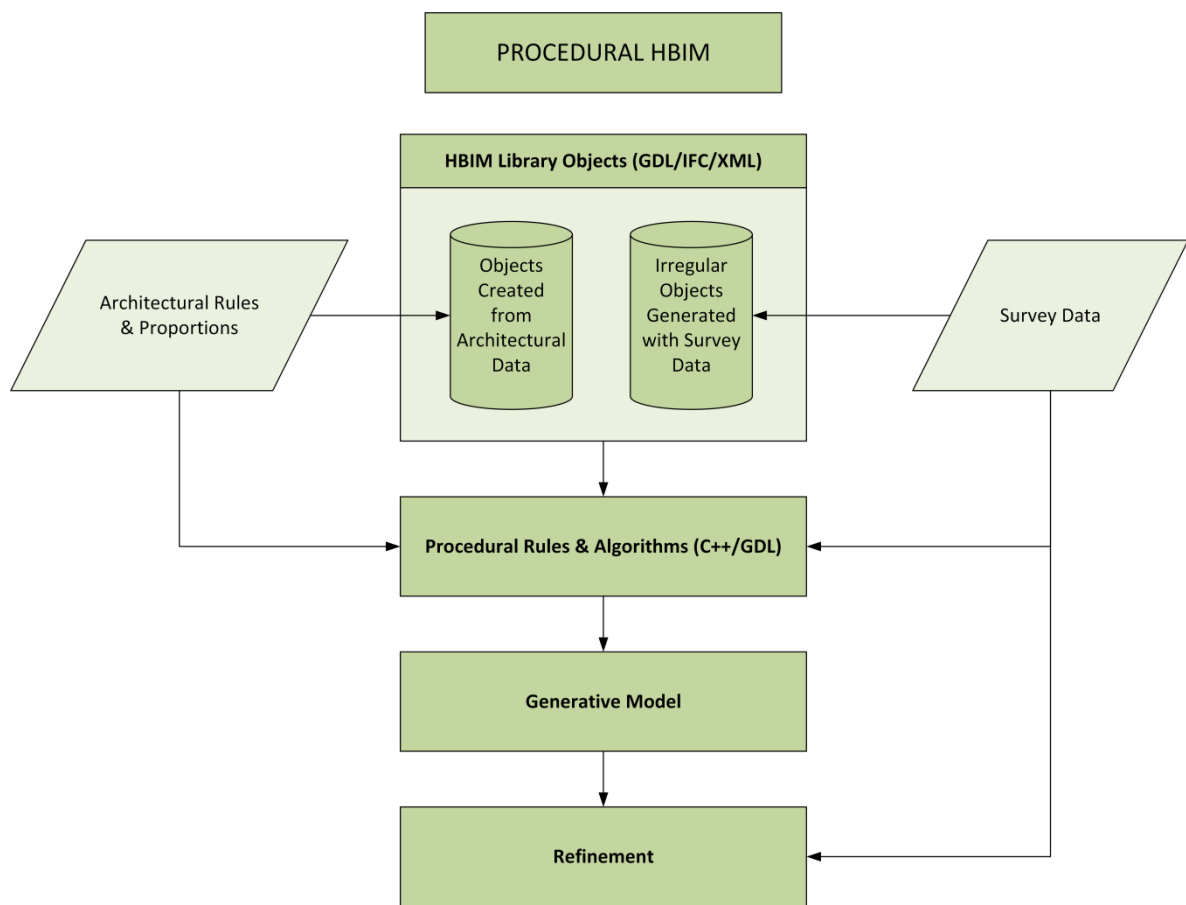


Figure 3.2: Workflow for new procedural HBIM approach.

The two main components of the new procedural HBIM approach (Figure 3.2) are a set of parametric library objects and a set of procedural rules and algorithms which automatically combine library objects to generate different building arrangements. Architectural rules and proportions and survey data are used as input data for both

parametric library objects and procedural rules. Once a model is generated with procedural modelling rules, survey data is used again for further refinement of computer generated models (Figure 3.2).

This chapter describes the design and initial implementation for the first of three procedural modelling prototypes developed for BIM software (Figure 3.1). This first prototype provides new procedural rules to generate building façade geometry. This chapter firstly contains an overview of the prototype followed by a description of the architectural rules used to assist with the reconstruction of façade geometry. The conceptual design framework for developing the procedural façade prototype is then described. Finally, the initial implementation and coding of this prototype is described. The application of the new prototypes with real case-studies is the full implementation which is later described in Chapter 7.

### **3.2 Overview of Procedural Building Façade**

The procedural façade prototype enables building facades to be modelled from survey data with a semi-automatic process. Procedural rules are used to generate different building façade arrangements which are controlled by altering parameters such as the number of storeys, number of horizontal tiles and door position. A generated façade model acts as a template for modelling many building façades by adjusting parameters. Parametric library objects such as windows and ashlar block wall detail can be selected and automatically added to building façade tiles. When automatically generating façade objects, the initial position and size of elements are estimated using classical architectural rules and proportions. The inclusion of architectural rules and proportions in the reconstruction process reduces the amount of further editing required. After a façade is automatically generated, users can then interactively edit the position and size of façade elements to accurately refine objects to survey data.

### 3.3 Architectural Rules for Procedural Façade

Similar to the design of the HBIM library objects in previous work (Murphy et al., 2013), the procedural rules for generating façade geometry also use architectural knowledge to assist with the digital reconstruction process. Architectural rules and proportions outlined in pattern books relating to classical building façades are used (Roche 1999). The proportioning of a classical façade is determined by the size and position of window openings, which can be expressed by the relationship between circles of the same radius (Figure 3.3). The top windows are made up of a single circle, in the next set of windows intersecting circles, and finally in the lower set of windows the circles are placed one on top of each other. Using a parameter for the window width or circle diameter the height and position of windows can be calculated with these proportions.

These proportions are evident in most classical buildings; however alterations to buildings can obscure or remove some of the original façade proportions. Alterations to a façade over time can include removal or enlarging of brick walls, window and door openings and parapets. After testing the classical proportions on a variety of surveyed façades (Figure 3.4) the most suitable proportions and parameters were adopted as seen in Figure 3.5. The parameter “*A*” represents window widths and the vertical distance between windows on the top floor and the floor below. The parameter “*B*” is used to represent horizontal and vertical window spacing on all other floors.



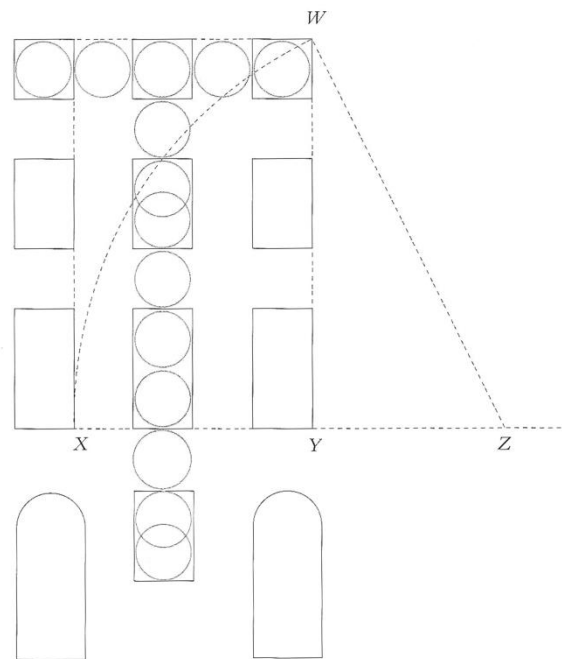


Figure 3.3: Proportions for façades and openings (Roche, 1999).



Figure 3.4: Testing classical proportions using orthographic images from surveyed classical buildings.

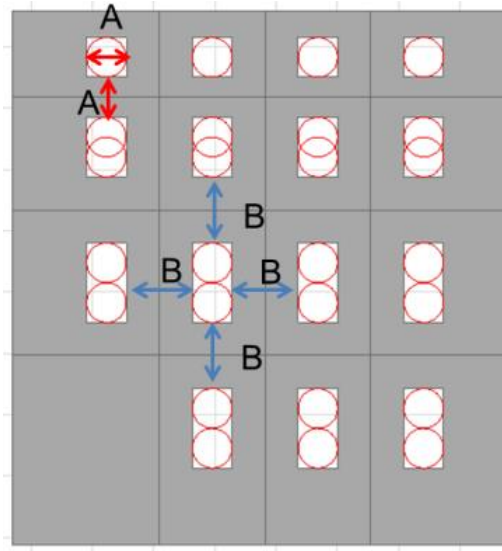


Figure 3.5: Proportions and parameters used for design of parametric façade.

### 3.4 Conceptual Design Framework

The conceptual design framework for the procedural façade prototype is based on concepts from shape grammars. A shape grammar is a production system used in procedural modelling to automatically generate two or three-dimensional geometries from a basic vocabulary of shapes and a set of production rules. A conceptual design framework based on parametric design and shape grammar principles is described in this section.

#### 3.4.1 Parametric Design

The first stage of design involved evaluating a set of key parameters that would allow a façade template to be modified to model many different building façade arrangements. Efficient methods for editing these parameters also had to be established. The standard method for editing parameters of a parametric object is to specify or edit parameters from a list in a dialogue box. This approach was adopted for modifying global parameters or parameters affecting the entire façade. Editing more specific parameters relating to an individual object such as a window opening would be very time-

consuming and inefficient using this method as it would require a large number of parameters to be measured and entered into a dialogue box. To facilitate efficient parameter editing another method using graphical parameter editing was designed. This method allows users to select a specific part of the façade in 2D or 3D and interactively edit the objects parameters by moving the object graphically. This enables parameters of the façade to be modified while overlaying the model with survey data in 3D or 2D. This removes the need for taking measurements and entering measurements into a dialogue box, instead, parameters can be matched to survey data directly in 3D or 2D. Figure 3.6 shows an example of a dialogue box for editing parameters from a list (left) and also graphical editing where a parameter for the distance between two floors is being altered in 3D (right).

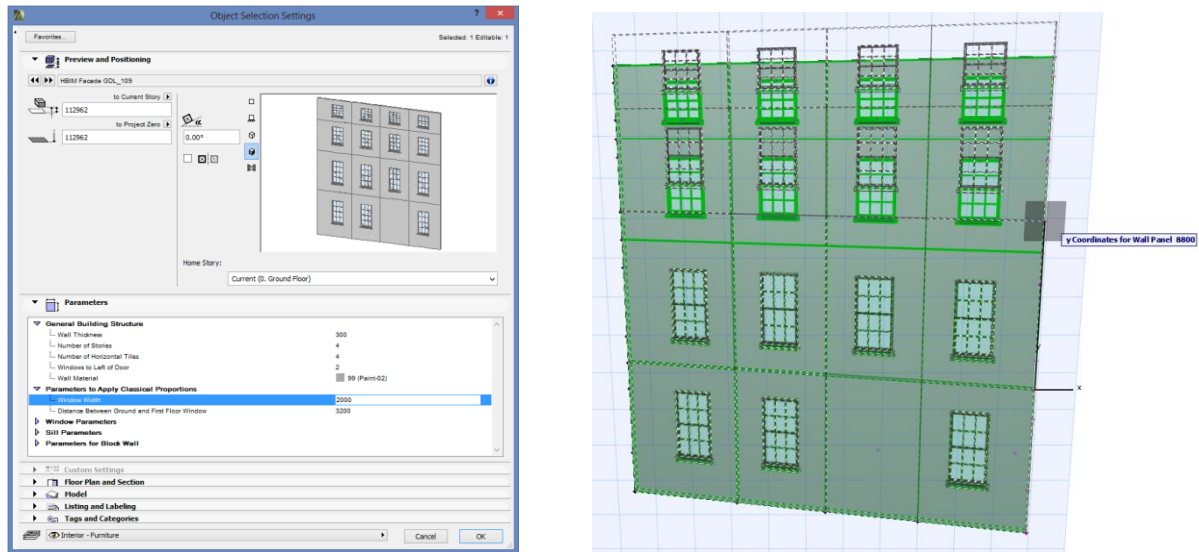


Figure 3.6: Modifying parameters from a dialogue box (left) and modifying parameters graphically (right)

The key parameters evaluated for modifying the façade template are shown in Table 3.1, along with the parameter type and methods of editing. The first group of parameters are designed to generate procedurally the structure of a façade. Parameters for the “Number of Storeys”, “Number of Horizontal Tiles”, “Door Position” and “Wall Thickness” are included in this group. These parameters can be edited graphically or

from a dialogue box. The second group of parameters are designed to calculate and apply classical architectural proportions outlined in section 3.3. These parameters apply a global update to the entire façade and must be entered from a dialogue box. The third group of parameters are designed for graphical editing only and enable any specific object or element on a façade to be altered. This includes graphically editing window corners, distances between floors, split lines between tiles and the overall façade width and height. Further architectural elements such as windows have additional parameters which can be altered from a dialogue box or by graphically editing specific instances or groups of instances.

Table 3.1: Key parameters designed for modifying the façade template

PARAMETER	EDITING METHOD	PARAMETER TYPE
<b>General Structure of Façade:</b>		
Number of Storeys	Dialogue Box & Graphical	Integer
Number of Horizontal Tiles	Dialogue Box & Graphical	Integer
Door Position (windows to left of door)	Dialogue Box & Graphical	Integer
Wall Thickness	Dialogue Box & Graphical	Real Number
<b>Parameters to Calculate Classical Proportions:</b>		
Window Width (Global)	Dialogue Box	Real Number
Distance Between Ground and First Floor Window Openings	Dialogue Box	Real Number
<b>Graphical Modifications Only:</b>		
Distance Between Floors (Tile Coordinates)	Graphical	Real Number (Array)
Distance Between Tiles (Tile Coordinates)	Graphical	Real Number (Array)
Building Width/Height (Tile Coordinates)	Graphical	Real Number (Array)
X,Y Coordinates of Individual Window Opening Corners	Graphical	Real Number (Array)

### 3.4.2 Shape Grammars

Concepts from shape grammars have been adapted to design and implement the procedural façade prototype. Shape grammars are a very suitable approach for architectural modelling as they allow models to be created from a vocabulary of basic shapes and set of replacement rules where a shape can be replaced or altered by

transformations, rules or algorithms. These principles facilitate the encoding of classical architectural rules and proportions which also use grammars or vocabularies for architectural elements and rules to combine these basic shapes.

In the field of computing, a standard shape grammar introduced by Stiny et al. (1972) can be defined as  $\{N, \Sigma, P, S\}$  where  $N$  and  $\Sigma$  represent a finite set of nonterminal and terminal shapes (the vocabulary),  $P$  are a set of production rules and  $S$  is a starting seed. Terminal shapes are the basic vocabulary elements and can be a collection of points, lines, planes, areas or solids. Non-terminal shapes are markers or boxes that are used to guide the terminal shapes during generation process and control the scope and position of shapes. The production process begins with nonterminal shapes which are replaced by terminal shapes when rules are applied. The production process terminates when no more rules can be applied and all nonterminal shapes have been removed. Production rules are applied in the form  $A \rightarrow B$  where  $A$  and  $B$  are nonterminal and terminal shapes. When a rule is applied the shape on the left-hand side is replaced by a new shape on the right-hand side of the rule.

Stiny (1977) also introduced the concept of a parametric shape grammar. This type of shape grammar differs from standard shape grammars in that it contains shape rules defined in terms of parameterized shapes. A parametric shape grammar can be defined as  $\{S, L, R, I, T\}$  where  $S$  is a finite set of shapes,  $L$  is a finite set of labelled points,  $R$  is a finite set of shape rules,  $I$  is an initial shape and  $T$  is a set of transformations. The main difference to the standard shape grammar is that nonterminal shapes are replaced by labelled points or parameters that are associated with shapes. Euclidean transformations have also been added which can include translations, rotations, scale or mirror. Shape rules are applied to a shape with an assignment of real values to the parameters and with additional transformations as required.

The main concept from shape grammars that have been adopted for developing the procedural façade prototype is the use of a basic shape vocabulary and shape rules. The shape rules are used to transform the shape vocabularies to automatically generate parametric façade geometry. The design for these shape grammar concepts is described in sections 3.4.3 below. The initial implementation of these shape grammar techniques with the Geometric Description Language (GDL) is described in Section 3.5.

### 3.4.3 Design of Parametric Shape Grammar Elements

The shape grammar techniques adopted for the design of the procedural façade incorporate the five elements of Stiny's Parametric Shape Grammar  $\{S, L, R, I, T\}$  (Stiny, 1977). The initial shape  $\{I\}$  is made up of a labelled solid shape as seen in Figure 3.7. This shape contains four labelled points as parameters which control the size and shape of the object. An additional parameter also defines a thickness which converts the 2D shape into a 3D solid shape.

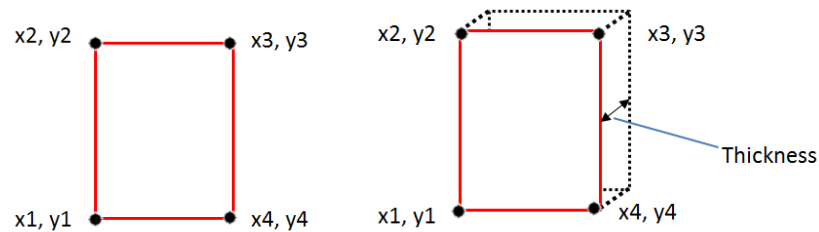


Figure 3.7: Initial shape  $\{I\}$  for parametric shape grammar design.

The basic elements that make up the vocabulary of shapes  $\{S\}$  can be seen in Figure 3.8. Shapes include two wall tiles; one wall tile  $TW$  which contains a window opening and surrounding wall. A second wall tile  $TD$  is used as a panel containing a door opening. Additional library objects relating to a door and door cases such as columns and pediments are linked with this shape  $TD$ . Other shapes include parametric library objects for windows ( $W$ ) and a simple block ( $BL$ ) that is used to create detail for ashlar

stone wall geometry. Additional library objects from previous work (Murphy et al. 2013) are also used in conjunction with these basic shapes in Figure 3.8.

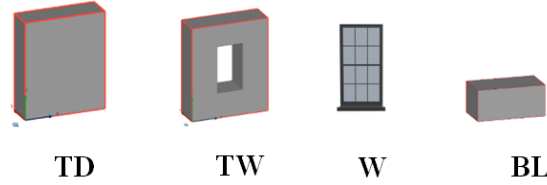


Figure 3.8: Basic shape vocabulary elements  $\{S\}$  for parametric shape grammar design.

The shape rules  $\{R\}$  used are shown in Table 3.2. Table 3.2 shows the initial shape on the left-hand side of each rule and the resulting shape on the right-hand side after the rule has been applied. Each rule is applied with an assignment of real values to parameters for shapes and transformations if required such as translations, rotations, scaling or mirroring.

**Rule 1:** replaces the initial shape  $\{I\}$  with the shape  $TW$  by adding an opening to the solid shape. The new shape  $TW$  has new parameters to define the coordinates of the opening.

**Rule 2:** repeats a shape along the x-axis. Parameters are used to control the number of repetitions or a distance which specifies the number of repetitions.

**Rule 3:** is similar to rule 2 but repeats shapes along the y-axis.

**Rule 4:** splits a shape along the x-axis into smaller or separate shapes. Parameters control the positions and number of splits.

**Rule 5:** is similar to rule 4 but splits shapes in the y-axis.

**Rule 6:** splits a shape in the x-axis and removes one of the segments.










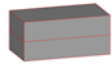





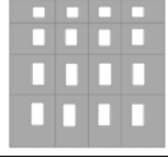




**Rule 7:** splits a shape in the y-axis and removes one of the segments.

**Rule 8:** conditional repeat (described below).

**Rule 9:** replaces a window tile  $TW$  with a door tile  $TD$ .

**Rule 10:** adds a selected window  $W$  from the library to a window tile  $TW$ .

Table 3.2: Shape rules  $\{R\}$  for parametric shape grammar design.

Rule	LHS		RHS	Description
Rule 1		$\longrightarrow$		Shape Replacement
Rule 2		$\longrightarrow$		Repeat x
Rule 3		$\longrightarrow$		Repeat y
Rule 4		$\longrightarrow$		Split x
Rule 5		$\longrightarrow$		Split y
Rule 6		$\longrightarrow$		Split x & Remove
Rule 7		$\longrightarrow$		Split y & Remove
Rule 8		$\longrightarrow$		Conditional Repeat x, y
Rule 9		$\longrightarrow$		Shape Replacement
Rule 10		$\longrightarrow$		Shape Replacement

Rule 8 is the main rule that creates a façade arrangement. This rule repeats the wall tile  $TW$  in both x and y directions based on various parameter settings and architectural rules. This method contrasts to the concepts adopted in the CGA shape grammar (Muller et al. 2006) where a façade is split into smaller tiles. This rule repeats the input shape in the y direction for the first column, and then moves to the second column and so on until all tiles have been placed for the specified parameters (Figure 3.9).



Parameters for each repeated instance of the shape *TW* are calculated based on architectural rules and the position of that instance in a façade arrangement. Parameters for the “number of floors” and “number of columns” control the number of repeated instances to be placed in each column and floor. The input for this rule is the shape *TW* and coordinates for this shape which are calculated and assigned as shapes parameters.

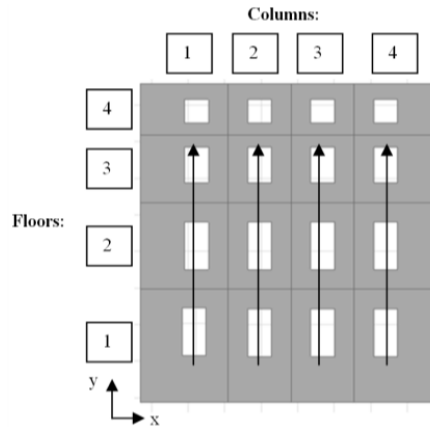


Figure 3.9: Repeated wall tile resulting from the application of rule 8 to shape *TW*.

Figure 3.10 shows the order and application of rules shown in Table 3.2 to create a basic façade arrangement. The application of rule 9 adds a door tile *TD*. The position of this is obtained from a user defined parameter “windows to left of door”. The application of rule 10 adds window objects to all window openings on the façade. Global parameters for window objects can be entered and set from the objects dialogue box. Specific parameters for a particular instance of a window object can be set using graphical parameter editing. Figure 3.11 shows another application of the rules in Table 3.2 used to create a parametric ashlar block wall that can be automatically combined with the parametric wall façade. Parameters allow the user to add ashlar block wall detail to the ground floor or all floors of the façade. Parameters of the block wall enable the user to change the individual block size, mortar spacing and texture.

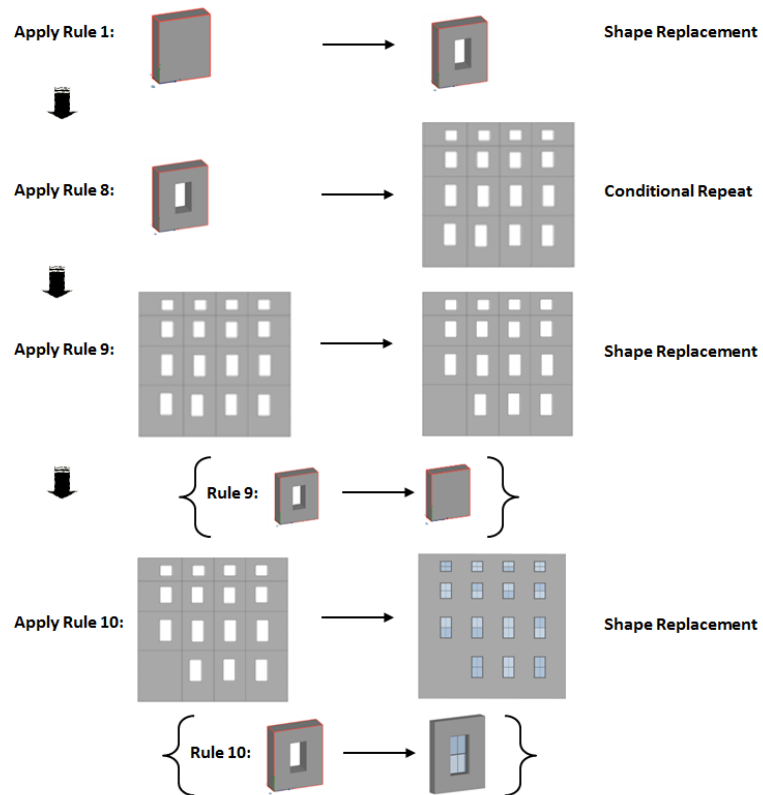


Figure 3.10: Application of shape rules specified in Table 3.2.

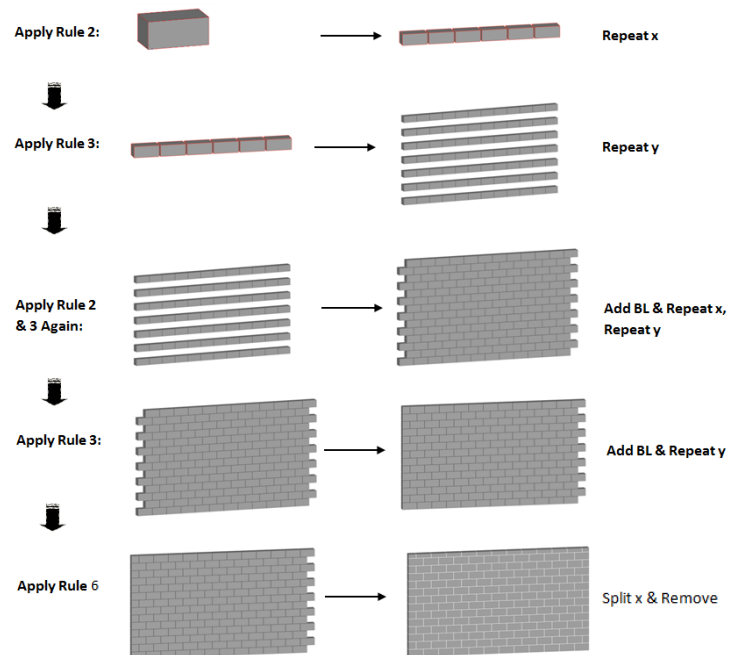


Figure 3.11: Another application of shape rules specified in Table 3.2 to create ashlar block wall detail.

### **3.5 Initial Implementation with the Geometric Description Language**

Various programs and languages have been used to implement shape grammars to generate shapes from a grammar. Examples include the CGA Shape grammar implemented in ESRI's CityEngine software (Muller et al., 2006). Other shape grammars have been implemented using the Generative Modelling Language (GML) (Hohmann et al., 2009) (Thaller et al., 2011). The concepts of a parametric shape grammar described in the previous section have been implemented using the Geometric Description Language (GDL), a programming language used for creating parametric objects within the ArchiCAD BIM software. The syntax of this language is similar to that of the BASIC programming language. GDL provides a large number of functions for creating 3D parametric objects using primitive shapes such as blocks, spheres, cones and ellipses or by generating shapes from 2D outlines. GDL uses coordinate transformation commands stored in a stack to position multiple objects relative to each other. GDL allows for graphical editing of parameters, complex Boolean operations, various control statements and the use of mathematical functions in creating parametric objects. Also provided is the ability to script a specific user interface for objects and their parameters (Graphisoft, 2011).

Although designed for parametric modelling, because of its powerful capabilities GDL can also be used to generate shapes using shape rules and shape vocabularies similar to a shape grammar. To date, this language has not been used to implement shape grammars. Using the GDL language with shape grammars enables the advantages of automated procedural modelling to be incorporated within a BIM environment.

### 3.5.1 Suitability of GDL for Shape Grammar Techniques

GDL structures content using executive scripts and subroutines. Parts of code can also be stored separately as macro objects which can be called from a script. This structure facilitates the implementation of a shape grammar as rules can be stored as individual macros which can be called from a script and applied to vocabulary shapes which are stored in individual subroutines. GDL also uses coordinate transformations stored in a stack to transform and position objects relative to each other. This includes the functions *ADD*, *ROT* and *MUL* which are used to apply translations, rotations and scaling. These transformations are very suitable for the Euclidean transformations which are part of the parametric shape grammar concept introduced by Stiny (1977).

Many procedural modelling techniques apply variation using rules chosen randomly along with random parameter assignment within rules. This allows for the creation of large scenes to be procedurally generated with a lot of variation. Within GDL this concept can be replicated using the *RND* function to generate a random value between defined constraints. This can be used to assign random parameter values and also to choose rules (stored as macro objects) to be applied randomly. Figure 3.12 shows an example of this where a short piece of code can automatically generate 3D content for a large scene (Figure 3.13) using random parameters and transformations values. This is created from one vocabulary shape (block) and one rule applied to this shape. A user parameter can control the number of iterations or repetitions of this shape.

### 3.5.2 Encoding Shape Grammar Techniques with GDL

Coding with the Geometric Description Language (GDL) is carried out using different scripts for different parts of the parametric object being created. The 3D script is the main script used to build the parametric 3D object. A 2D script is used to program how the object will be represented in plan view. A master script is the first script that is

executed and is used for tasks such as reading in value lists, setting up parameters, checking user errors and defining materials. Information in the master script can be used in all other scripts including the 2D and 3D scripts.

```

1  FOR i=1 TO number_of_iterations
2      PARAMETERS L[i] = RND (5000),
3      W[i] = RND (5000),
4      H[i] = RND (10000)
5
6  NEXT i
7
8  FOR K=1 TO number_of_iterations
9      FOR p=1 TO number_of_iterations
10         BLOCK L[p],W[p],H[p]
11         ADDx (L[k])
12         ADDy (RND (10000))
13         ADDy -(RND (10000))
14     NEXT p
15     DEL TOP
16 NEXT K

```

Creates an array of random parameters (between defined limits) to be used for BLOCK primitive.

Block using array parameters defined above.

Transformations stored in stack to position/move elements.

Figure 3.12: GDL code which randomly generates 3D content shown in Figure 3.13.

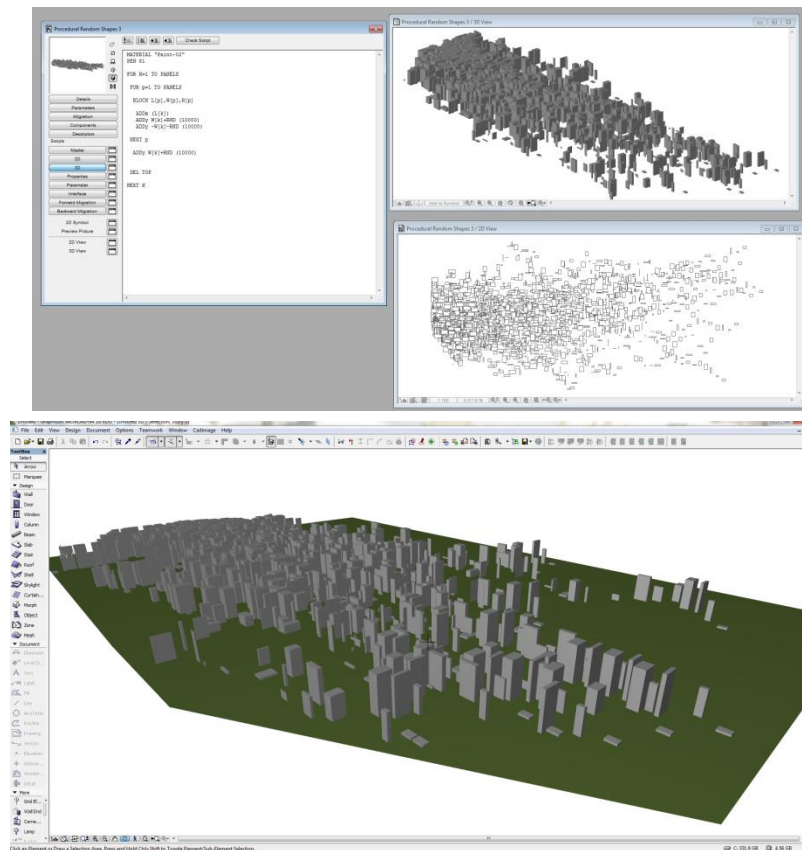


Figure 3.13: Automatically generated 3D content with GDL using random parameters and transformations.

A parameter table is used to define the parameters for the object. Various different parameter types are available for parameters including real numbers, integers, Boolean values and text. Parameters can also be stored in arrays. A separate parameter script is used to create drop-down value lists, set up ranges or constraints for parameters and locking or altering parameters. A property script is used to write components and descriptor commands if the object is to be used in a schedule. Finally, a user interface script enables a custom dialogue box to be created with custom text fields, images, user interface buttons and input fields. The different scripts used to create the parametric façade are described below.

### 3.5.3 3D Script

The main shape vocabulary objects and rules developed for the parametric façade have been coded in the 3D script. The 3D script is structured with subroutines and an executive script which is a controlling script used to call subroutines. Shape vocabulary objects are stored in individual subroutines and shape rules which are applied to shape vocabulary objects are coded in the executive script. The initial shape  $\{I\}$  (Figure 3.7) used as part of the parametric shape grammar is coded using a GDL function *cPRSIM* and stored in a subroutine. This is coded by defining the shape using coordinates on the x-y plane and specifying a height or thickness. The shape can be rotated using the transformation command *ROT* to lift it off the x-y plane. The coordinates are input as parameters which can alter the shape and size of the object. This same command is used to create the vocabulary shapes *TW*, *TD* and *BL* (Figure 3.8) which are all stored in individual subroutines. Figure 3.14 shows the GDL code for the shape *TW* which is called when a rule is being applied to this shape. This code is similar to the code for the initial shape  $\{I\}$  but includes coordinates for an opening which is used for window openings in the wall façade.

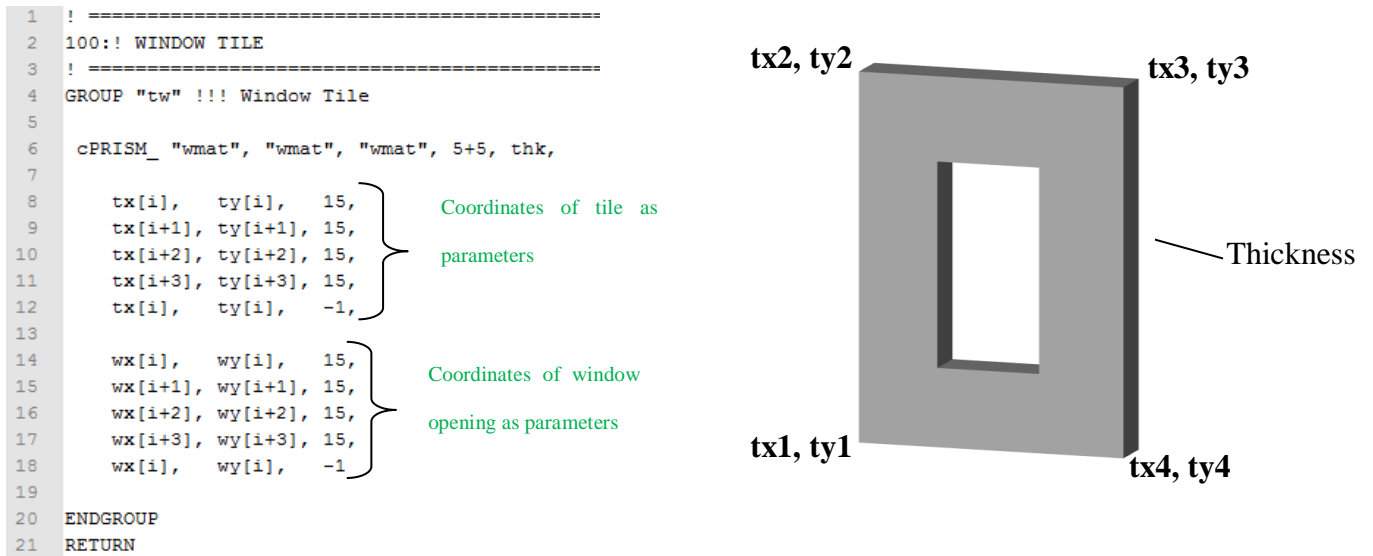


Figure 3.14: GDL code for vocabulary shape *TW*.

Other shape vocabulary objects stored in subroutines include a parametric sash window. This parametric sash window is the default window for the parametric façade. The default sash window created for the parametric façade has parameters to alter the window width, height, width and thickness of the window frame, sash frame and glazing bars, reveal depth and the number of vertical and horizontal glass panes in each sash. Sample code for this vocabulary object can be seen in Figure 3.15.

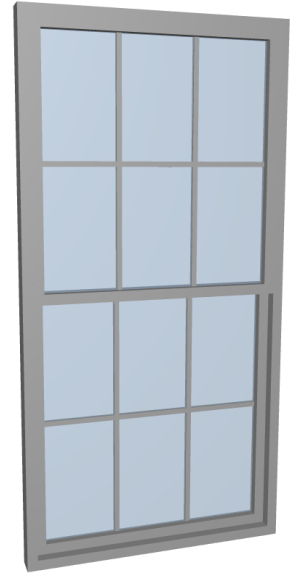
Along with the window object is a separate object for a window sill (Figure 3.16). Parameters for the sill include the sill front height, sill angle, sill back height, spacing between sill and wall, sill nosing and sill overhang at left and right. Another vocabulary object used in the parametric façade is an ashlar block wall (Figure 3.17). This is created from a basic block and the application of shape rules outlined in Figure 3.9. Using GDL these rules are coded using loops to repeat a block in the x-direction and then repeat this row in the y-direction. This is repeated for alternating rows. Geometry is also added for mortar between blocks. The GDL function *cPRISM* is used for this. Parameters for this vocabulary object include individual block length, height and

thickness, block wall width and height and mortar spacing. Sample code for this object can be seen in Figure 3.17.

```

1  ! =====
2  ! 400:!! SASH WINDOW:
3  ! =====
4  !
5  !
6  ! =====
7  ! Executive Script for Sash Window:
8  ! =====
9
10 MATERIAL frame_mat
11 PEN win_pen
12
13 GOSUB 410: !!! Window Frame
14   ADD frame_width, frame_width, 0
15   ADDy lower_sash_opening[i]
16 GOSUB 420: !!! Lower Sash Frame
17   DEL 1
18
19   ADD 0, lower_sash_box_height[i]-sash_frame_width, sash_frame_thk
20   ADDy -upper_sash_opening[i]
21 GOSUB 430: !!! Upper Sash Frame
22   DEL 3
23
24 ! =====
25 ! !!! Lower Glazing Bars
26 ! =====
27
28   ADD frame_width+sash_frame_width, frame_width+sash_frame_width, 0
29   ADDy lower_sash_opening[i]
30
31 FOR iii=1 TO (hor_lower[i]-1)
32   ADDx lgx[i][iii]
33   GOSUB 470: !!! Vertical Glazing Bar Lower
34   DEL 1
35 NEXT iii

```



W

Figure 3.15: Sample script for parametric sash window stored as a vocabulary shape in a subroutine.

```

1  ! =====
2  490:!! Sill
3  ! =====
4  GROUP "Sill"
5
6 PARAMETERS tri_height = (sin(sill_angle)*(sill_width-sill_top_width))/sin(180-(90+sill_angle))
7 MATERIAL sill_mat
8
9   ROTy -90
10  ROTy 180
11  ADDx -sill_width
12
13  cPRISM_ "sill_mat", "sill_mat", "sill_mat", 7, sill_depth,
14
15    0, 0, 15,
16    0, sill_height_front, 15,
17    (sill_width-sill_top_width), sill_height_front+tri_height, 15,
18    sill_width-sill_top_width, sill_height_back, 15,
19    sill_width, sill_height_back, 15,
20    sill_width, 0, 15,
21    0, 0, -1
22
23  DEL 3
24
25 ENGROUP
26 RETURN
27

```



S

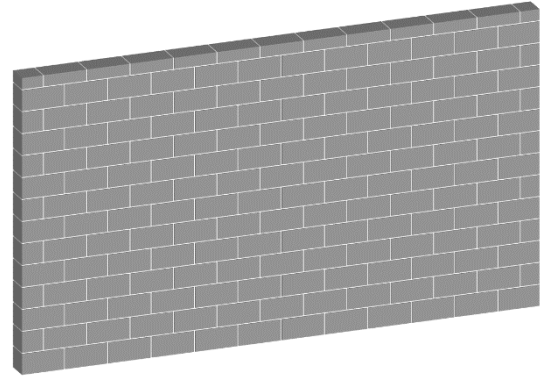
Figure 3.16: Script for a parametric sill stored as a vocabulary shape in a subroutine.



```

1
2 ! =====
3 300:! Block Wall (Ground Storey)
4 ! =====
5
6 ! =====
7 ! Executive Script for Block Wall:
8 ! =====
9
10 GROUP "BlockWall"
11
12 ADD px[1],py[1],0
13
14 FOR ht_i = 0 TO wall_ht[1] STEP ((bl_space + by[2])*2)
15
16 ADDy ht_i
17
18 FOR wall_len_i = 0 TO wall_len STEP (bl_space + bx[4])
19
20 ADDx wall_len_i
21 GOSUB 350:
22 DEL 1
23
24 NEXT wall_len_i
25
26 DEL 1
27
28 NEXT ht_i
29
30 !!! LOOPS FOR INDENTED BLOCKS (even rows):
31
32 ADDx bx[3]/2+bl_space
33 ADDy (bl_space + by[2])
34
35 FOR ht_i = 0 TO wall_ht[1] STEP ((bl_space + by[2])*2)
36

```



**BW**

Figure 3.17: Sample script for ashlar block wall stored as a vocabulary shape in a subroutine.

These various vocabulary shapes are combined and repeated using shape rules to create a parametric façade that can be used to automatically generate endless configurations of building façades. The shape rules described in section 3.4.3 are also coded in the 3D script but are located at the top of the 3D script in an executive script. When a rule is applied to a shape vocabulary object the required object is called from its subroutine and altered by the rule.

Rules two and three (“Repeat x” and “Repeat y”) from Table 3.2 are implemented in GDL using loop commands to repeat shapes in a certain direction. The GDL functions *CUTPLANE* and *CUTPOLY* are used for rules four and five to split geometry into multiple components. Rules six and seven use Boolean operations in GDL such as *SUBTRACT* to remove segments after splitting.

The main rule that is used to generate the structure of a façade is rule 8 (Table 3.2), which repeats the vocabulary shape *TW* in both x and y directions as outlined in section

3.4.3. This is coded in GDL with a loop command to repeat the shape *TW*. When a parametric shape is normally repeated in a loop the parameter values of the repeated instances will match that of the first instance. So if a parameter value is changed for the object it will also change in all repeated instances. This is not suitable for repeating the vocabulary shape *TW* for the parametric façade as repeated instances will require different parameter values on each iteration to allow for different parameter values such as window heights and widths throughout a façade. With GDL arrays are used to overcome this problem. Specifying a parameter as an array in the parameter table allows a single parameter to have multiple values stored in a table (Figure 3.18). When the shape *TW* is repeated in a loop with rule eight a different parameter value from an array are assigned to each repeated instance. This allows each repeated instance of the shape to have unique parameter assignments each time.

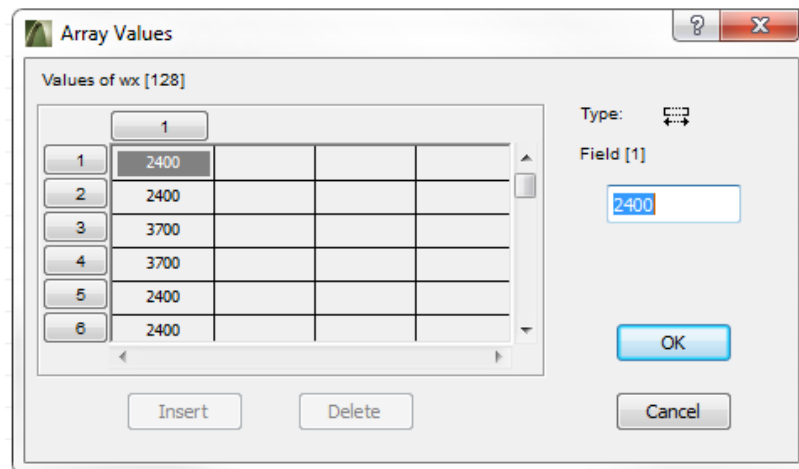


Figure 3.18: Parameters as an array to allow multiple assignments for a parameter.

While repeating the shape *TW* using a loop command, *IF* statements are used to control the number of repeated instances in a particular column and also the number of columns to be placed (Figure 3.9). The user parameters for “number of floors” and “number of horizontal tiles” are used to control this. Coordinate transformations are also used with this loop command to control where the next instance is to be placed. After the first

instance is placed a coordinate transformation is used to move to the next position for the second instance. After all instances have been placed on a particular column then a coordinate transformation moves to the bottom of the next column for the next instance and so on until all instances have been placed for the specified parameter settings. Sample code for these coordinate transformations embedded in a loop as part of rule eight can be seen in Figure 3.19.

```

1  ! -----
2  !!! IF STATEMENTS TO MOVE ORIGIN DURING LOOP:
3  ! -----
4
5  IF i=5 OR i=21 OR i=37 OR i=53 OR i=69 OR i=85 OR i=101 OR i=117 THEN
6      ADDy py[2]
7  ENDIF
8
9  IF i=9 OR i=25 OR i=41 OR i=57 OR i=73 OR i=89 OR i=105 OR i=121 THEN
10     ADDy py[6]
11 ENDIF
12
13 IF i=13 AND H=2 OR i=29 AND H=2 OR i=45 AND H=2 OR i=61 AND H=2 OR i=77 AND H=2 OR i=93 AND H=2 OR i=109 AND H=2 OR i=125 AND H=2 THEN
14     ADDy py[2]
15 ENDIF
16
17 IF i=13 AND H=3 OR i=29 AND H=3 OR i=45 AND H=3 OR i=61 AND H=3 OR i=77 AND H=3 OR i=93 AND H=3 OR i=109 AND H=3 OR i=125 AND H=3 THEN
18     ADDy py[6]
19 ENDIF
20
21 IF i=13 AND H=4 OR i=29 AND H=4 OR i=45 AND H=4 OR i=61 AND H=4 OR i=77 AND H=4 OR i=93 AND H=4 OR i=109 AND H=4 OR i=125 AND H=4 THEN
22     ADDy py[10]
23 ENDIF
24
25 FOR oi=17 TO 113 STEP 16
26     IF i=oi AND H=2 THEN
27         ADDx px[i-13]
28         ADDy -py[i-15]

```

Figure 3.19: Sample code showing coordinate transformations embedded in a loop as part of rule eight (Table 3.2).

Rule nine from Table 3.2 is used to place a door tile (*TD*) into the generated facade. The position of a door tile is obtained from the user defined parameter “Windows to left of Door”. Rule ten from Table 3.2 is used to add the vocabulary shape for a window object (*W*) to existing window tiles (*TW*) as shown in Figure 3.20. Parameters for the parametric window object are again stored in arrays to allow for different parameter assignments for each repeated instance. The width and height of each repeated window object are automatically calculated from the size of the window openings in each window tile and automatically entered into the relevant arrays for the window object.

When the window object is repeated throughout the façade the correct parameters for its width and height are automatically entered from the relevant arrays.

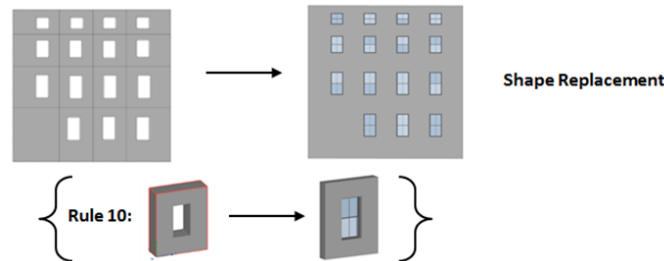


Figure 3.20: Shape rule 10 which adds a window object (*W*) to all existing window tiles (*TW*).

Another rule is also used to add the vocabulary shape for a parametric ashlar block wall (Figure 3.17) to the generated façade. The parameters for the width and height of the ashlar block wall are automatically matched to the parameters of the parametric façade. Users can choose to add ashlar block wall detail to all storeys of the generated façade or just the ground floor. When the ashlar block wall is being added to the façade, Boolean operations are used to cut areas for window openings in the wall. Figure 3.21 shows an example of this vocabulary shape automatically added to the generated façade.

Also included in the 3D script is code which defines how parts of the façade can be moved or graphically edited. This facilitates more efficient parameter editing as all objects don't have to be modified by altering parameter values in a dialogue box. Instead, parameters can be altered by graphically editing the objects in 2D or 3D. In GDL graphical hotspots are used to control how parts of the object can be moved. Graphical editing points are defined in the script and shown on the model with purple markers as shown in Figure 3.22. Clicking on a marker allows a certain parameter to be edited by moving the marker to a new position as seen in Figure 3.22 where the width of the façade is being graphically edited. The coding of this in GDL requires the editing

point to be defined using coordinates along with a vector for movement and the parameter being edited by the hotspot (Figure 3.23).

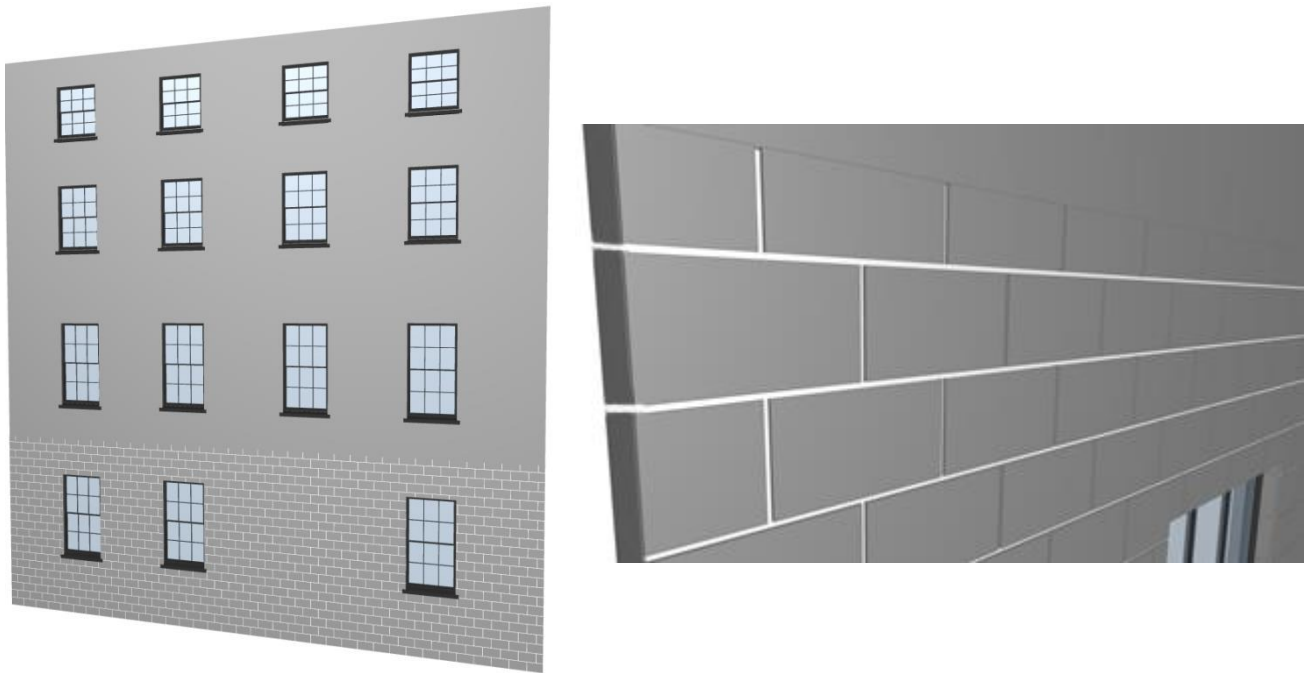


Figure 3.21: Parametric façade showing block wall detail automatically added.

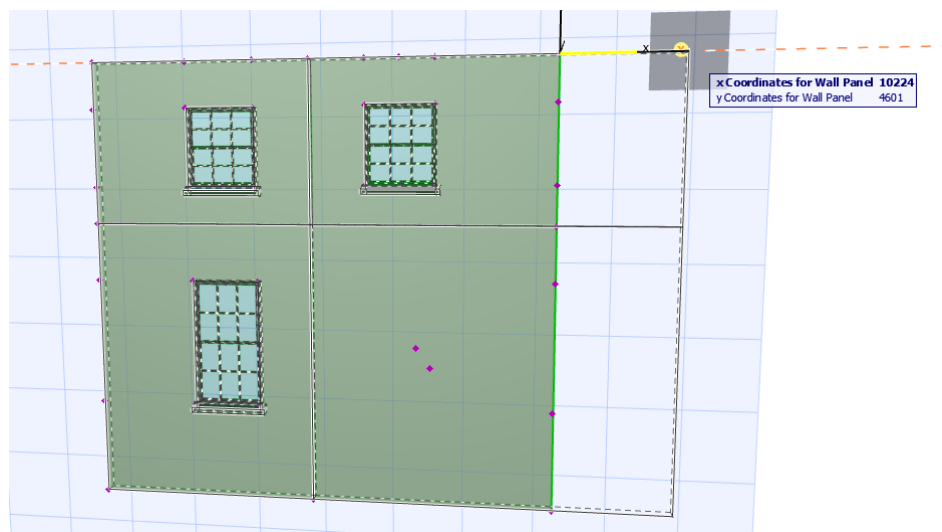


Figure 3.22: Graphical editing of the façade width. Graphical editing points shown with purple markers.

```

1
2 !!! Hotspots to edit lower right wall corner(x-coordinate):
3
4 IF i=13 THEN
5
6 HOTSPOT 0,      py[i+1], G, unID, px[1], 1+128: unID=unID+1 !Base Hotspot (Hidden)
7 HOTSPOT px[i+1], py[i+1], G, unID, px[1], 2:      unID=unID+1 !Moving Hotspot
8 HOTSPOT -px[i+2], py[i], G, unID, px[1], 3:      unID=unID+1 !Reference Hotspot (Hidden)
9
10 ENDIF
11
12
13
14 !!! Hotspots to edit lower right wall corner (y-coordinate):
15
16 IF i=113 THEN
17
18 HOTSPOT px[i+3], 0,      G, unID, py[i-112], 1+128: unID=unID+1 !Base Hotspot (Hidden)
19 HOTSPOT px[i+3], py[i+3], G, unID, py[i-112], 2:      unID=unID+1 !Moving Hotspot
20 HOTSPOT px[i+3], -py[i+2], G, unID, py[i-112], 3:      unID=unID+1 !Reference Hotspot (Hidden)
21
22 ENDIF

```

Figure 3.23: Sample GDL code for graphical parameter editing.

### 3.5.4 Master Script

A master script has been used to set up parameters, populate parameter arrays, define global parametric expressions and define material for objects. The classical proportions outlined in section 3.3 are set up and coded in the master script. These proportions provide an initial estimate for the position and size of elements on the façade which can be later edited to accurately match survey data. These proportions are calculated from two user defined parameters for “window width” and “distance between ground and first-floor windows”. A set of parametric expressions calculate the classical proportions from the two user defined parameters. Sample GDL code for these parametric expressions can be seen in Figure 3.24. After these proportions are calculated the coordinates of each tile and window opening are returned and stored in four arrays. When the shape *TW* is repeated with rule 8, the coordinates for the window tile (*TW*) in each iteration are obtained from these four arrays resulting in a façade with classically proportioned window openings as seen in Figure 3.21.

Another feature of the developed parametric façade is the ability to graphically edit parameters for a group of objects simultaneously. This allows for very quick editing of elements on the façade. For example, it is possible to edit the height of all windows on a floor simultaneously along with separate editing of individual window heights. The

coding to allow for editing groups of objects together is implemented in the master script. Normally graphically editing multiple parameters is not possible with GDL as coding an editable hotspot only allows one parameter to be edited at a time. To overcome this, a single parameter is used with a graphical hotspot which when altered applies a change to multiple other parameters. This allows one parameter to be graphically edited which updates multiple other parameters simultaneously in a group such as changing the height of all windows on a floor or the width of all windows in a column. Sample GDL code for this is shown in Figure 3.25.

```

344  !!! x-coordinates for 1st set of vertical panels:
345
346  FOR i=1 TO 16 STEP 4
347
348      PARAMETERS px[i+2] = C+AA+BB/2,
349                  px[i+3] = C+AA+BB/2
350
351  NEXT i
352
353  !!! x-coordinates for 2nd and 3rd to final set of vertical panels:
354
355  FOR i=17 TO 111 STEP 4
356
357      PARAMETERS px[i+2] = BB+AA,
358                  px[i+3] = BB+AA
359
360  NEXT i
361
362  !!! x coordinates for final set of vertical panels:
363
364  FOR i=113 TO 128 STEP 4
365
366      PARAMETERS px[i+2] = BB/2+AA+D,
367                  px[i+3] = BB/2+AA+D
368

```

Figure 3.24: Sample GDL code showing parametric expressions used to define classical proportions.

### 3.5.5 Other Scripts

A parameter script has been used for the parametric façade to specify drop-down values for parameters, hiding parameters that are not required and also specifying parameter constraints and allowable ranges. Parameter constraints and ranges are used to restrict users from entering incorrect parameter values or graphically editing parameters that may cause the model to degenerate. For example, a parameter for a window width

should not be negative and should not be larger than the wall tile that the window is placed in.

A 2D script is also used with the parametric façade to specify how the object will appear in 2D. This 2D representation can be coded to define a specific plan view or a command can be used to automatically generate lines for the 2D representation which is a projection of the object defined in the 3D script. This command has been used for the parametric façade to automatically project the plan view from the object created in the 3D script. Figure 3.26 shows examples of various façade arrangements automatically generated with the new procedural façade rules. A video showing a demonstration of the new procedural façade prototype can also be seen from the link below.

**<https://youtu.be/NujkDfN01Ig>**

```
1  ! =====
2  !!!Loops declaring addition parameters and join to relative window parameter for simultaneous editing of windows
3  ! =====
4
5  IF HS = 1 THEN
6
7    IF right_angle_opening= 0 THEN
8
9      FOR it=1 to 13 STEP 4
10
11        FOR iteration=it TO 128 STEP 16
12          wy[iteration] =  additiony[it]  + ABS (wy[iteration])
13          wy[iteration+1] = additiony[it+1] + ABS (wy[iteration+1])
14          wy[iteration+2] = additiony[it+1] + ABS (wy[iteration+2])
15          wy[iteration+3] = additiony[it]  + ABS (wy[iteration+3])
16
17        NEXT iteration
18
19      NEXT it
20
21    FOR it=1 to 125 STEP 16
22
23      FOR iteration=it TO (it+12) STEP 4
24        wx[iteration] =  additionx[it+13] + ABS (wx[iteration])
25        wx[iteration+1] = additionx[it+13] + ABS (wx[iteration+1])
26        wx[iteration+2] = additionx[it+14] + ABS (wx[iteration+2])
27        wx[iteration+3] = additionx[it+14] + ABS (wx[iteration+3])
28
```

Figure 3.25: Sample GDL code used to alter groups of parameters for simultaneously editing.



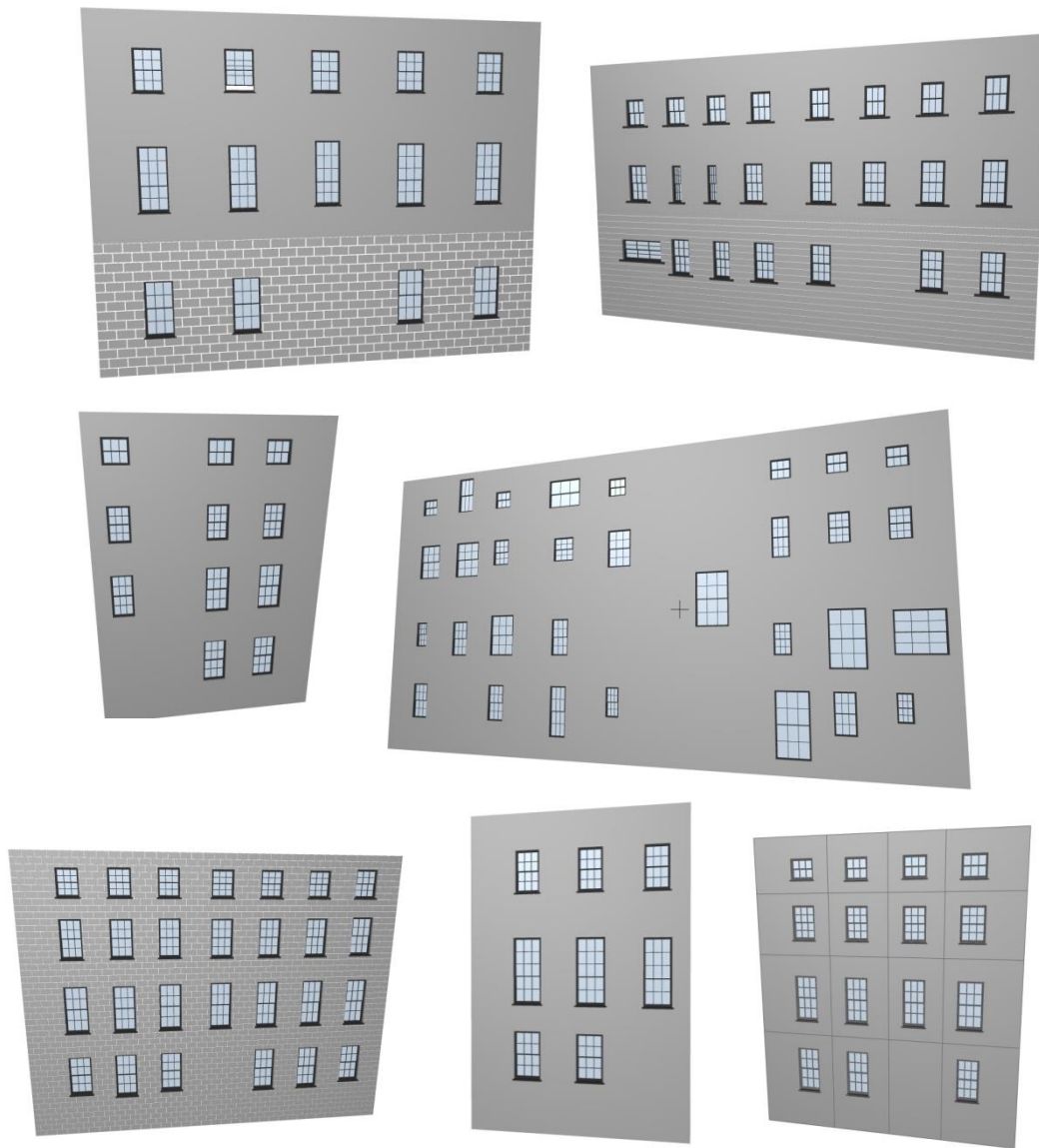


Figure 3.26: Various façade models automatically generated by altering parameters of the new procedural façade template.

### 3.5.6 Non-Uniform and Irregular Geometries

One of the main challenges involved with modelling existing buildings is accurately representing the current condition of a building which may include damage or deformation to a building or building parts caused over time. Damage caused by environmental conditions, settlement of the building on the ground or other causes may result in irregular or non-uniform geometries such as walls not intersecting exactly at 90-degree angles or non-vertical walls. Creating an accurate ‘as-is’ representation of an existing building should include these irregularities to ensure an accurate representation

of the building. Most architectural modelling software (including BIM software) is focused on new building designs and as a result, has very limited tools for modelling irregular and non-uniform building elements. In order to accurately model existing buildings, the new procedural façade rules were developed with options to create non-uniform and irregular geometries. This includes parameters to specify the inclination angle for non-vertical walls, options to specify exact corners for window openings which do not have to be perfect 90-degree rectangles. Windows automatically placed in these window openings are adjusted with irregular window frames to fit irregular wall openings. Newly developed parametric sash window objects also include options for specifying irregular grid positions for glazing bars.

### **3.6 Summary**

This chapter has outlined the steps involved in designing and implementing a new procedural façade prototype as a plug-in for the ArchiCAD BIM software. Architectural knowledge is used to assist with the reconstruction process by providing an initial estimate for the position and size of façade elements. The conceptual design framework for this parametric façade incorporates techniques from shape grammars and parametric design. This conceptual design framework was implemented and coded with the Geometric Description Language (GDL), an embedded programming language within the ArchiCAD BIM software. This enabled the tools to be used in a BIM environment. The process for refining the generated façade geometry to survey data is described in more detail in Chapter 6.

### **3.7 Limitations of Prototype**

The developed procedural façade prototype can greatly improve the efficiency of modelling existing building façades by automatically generating façade arrangements

and automatically generating objects on these façade arrangements. However, there are a number of limitations with the implementation of this initial prototype. Due to the design of the procedural façade rules, resulting models have limited scope for variation. For example façade models generated with these rules are limited to façades with two, three or four storeys while the range for horizontal tiles is limited between two and eight horizontal tiles. This is due to the design as transformations required for different arrangements are not extendable and instead coded for a limited number of different variations.

While the Geometric Description Language (GDL) is very suitable for developing procedural rules it also has its limitations. Plug-ins created for ArchiCAD which are implemented with only GDL have limited capabilities for interacting with existing ArchiCAD tools and functionality. For example, models created from GDL scripts are standalone objects and cannot interact or be altered by other ArchiCAD objects. A user-interface for a prototype implemented with GDL is also limited to the dialogue box for a GDL object. A plug-in containing a lot of functionality may require a more comprehensive user interface with toolbars, progress windows, unique dialogue boxes or integrated menu commands.

A design and implementation for a new more comprehensive procedural modelling prototype is described in chapter 4 which overcomes the limitations of this first procedural modelling prototype. A better design is developed to avoid previous limitations with the scope of variation. Problems with the previous implementation are also overcome by utilising both GDL and C++ programming languages combined with an application programming interface (API) for ArchiCAD software. This enables better integration with existing tools and functionality of ArchiCAD along with an improved graphical user-interface.

## Chapter Four: Prototype II – Procedural Building

### 4.1 Introduction

Two additional prototypes have been developed in order to overcome limitations of the first procedural façade prototype. These new prototypes provide more advanced capabilities for modelling historic buildings. These new prototypes enable more efficient modelling workflows as survey data can be used as input to the procedural rules for faster generation of building geometry. This chapter describes the conceptual design framework and initial implementation for Prototype II (Figure 4.1). This prototype further extends the concepts of the first procedural façade prototype by providing procedural modelling capabilities for all building faces and not just one façade.

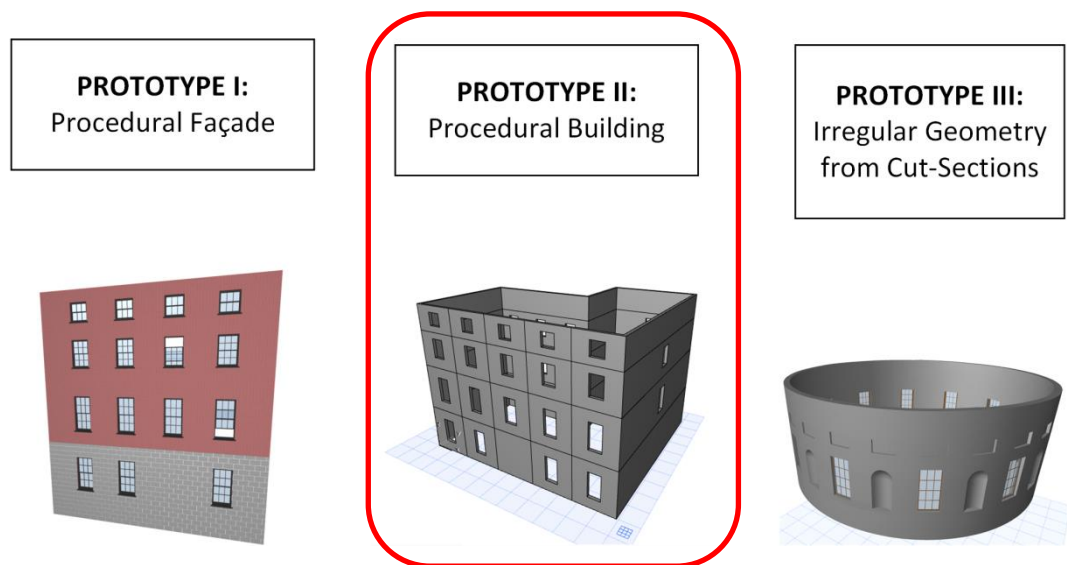


Figure 4.1: Procedural Modelling Prototypes for generating building geometry.

### 4.2 Overview of Prototype II: Procedural Building

Similar to the first façade prototype, the second prototype enables modelling existing buildings with a semi-automatic process where the required geometry is first generated

and then manually refined to match to specific survey data. Unlike the first prototype, this second prototype provides procedural rules for modelling all faces of a building and not just a single façade. This requires procedural rules capable of generating many types of building shapes. To achieve this, procedural rules are designed to interact with existing 2D building footprints which enables the automatic generation of any building shape. After the structure of a building is generated other procedural rules can then be applied to split a building face into tiles and automatically add building components. Generated geometry can then be manually refined to survey data. This prototype has again been implemented for the ArchiCAD BIM software using the Geometric Description Language (GDL). The C++ programming language has also been used for the implementation of this prototype with an Application Programming Interface (API) for the ArchiCAD software. This enabled the development of more advanced functionality that can interact with existing ArchiCAD tools.

### **4.3 Prototype II: Rule and Algorithmic Design**

The conceptual design framework for the procedural building prototype is also based on concepts from shape grammars where a vocabulary of shapes is used with a set of rules and algorithms to automatically generate different building arrangements. Similar classical proportions described in the section 3.3 are used to provide an initial estimate for the position and size of building elements on a generated building (Figure 3.3). The design for the procedural building prototype incorporates similar vocabulary shapes as the previous procedural façade (Figure 3.8) but with additional new objects such as a floor slab with joists (Figure 4.32). A new set of procedural rules and algorithms have been designed which are described in this section. Section 4.4 then describes the initial implementation of these new procedural rules.

#### 4.3.1 Rule One: Procedural Extrusion

The first procedural rule is used to extrude a building footprint (*FP*) or user drawn 2D polygon to create a mass model (*MM*) for a particular floor or building (Figure 4.2). This allows the procedural modelling techniques and rules developed to be applied to any building shapes. The volumetric mass model can be automatically generated from an existing building footprint or a user can define the footprint by drawing a new 2D polygon. This rule can be applied to any closed polygon which can contain both lines and arcs. The rule will automatically generate a volumetric mass model that can contain planar and curved surfaces. The height at which the building footprint is extruded to is a parameter of this rule.

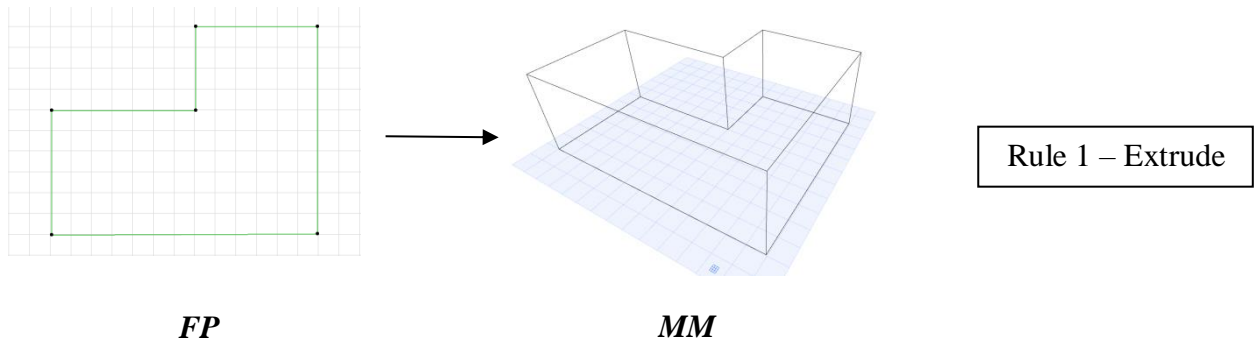


Figure 4.2: Rule One – Extrude a building footprint or user-drawn 2D polygon to create a mass model

The input for this rule is 2D polygon data. Users can select a single building footprint or multiple building footprints (*FP*) to automatically generate any number of building models at once. The input data automatically extracted from selected polygons include the number of selected polygons, number of sides per polygon, x and y coordinates of polygon nodes, number of arcs in each polygon (if any), x and y coordinates of arc start and end points and arc angles in radians (Figure 4.3).

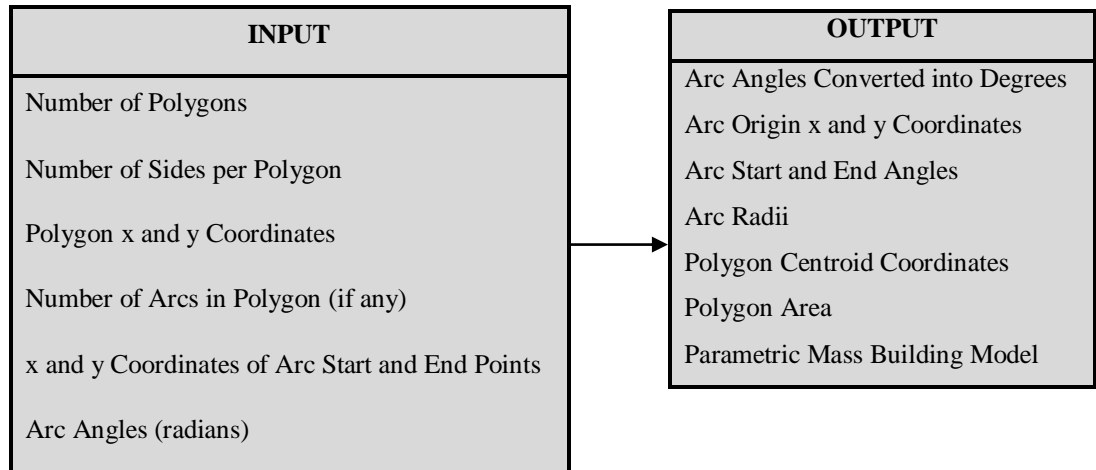


Figure 4.3: Inputs and outputs from procedural rule one – Procedural Extrusion

Figure 4.4 shows a workflow diagram for the steps involved in this first procedural rule. If polygons contain arcs then a number of initial calculations are performed to acquire data relating to arcs for later operations. As all arc angles extracted from polygons are in radians, they first need to be converted to degrees using the formula in Equation 4.1. Creating building mass models with GDL requires the origin of arcs to be calculated. This is calculated from the arc start and end coordinates and arc angle using the steps in Table 4.1 and the formulae in Equation 4.2 to Equation 4.7. Next, the angles from the arc origins to arc start and end points are calculated using Equation 4.3. These angles are stored for later operations with subsequent procedural rules. The radii of arcs are also calculated using Equation 4.2 and stored for later operations.

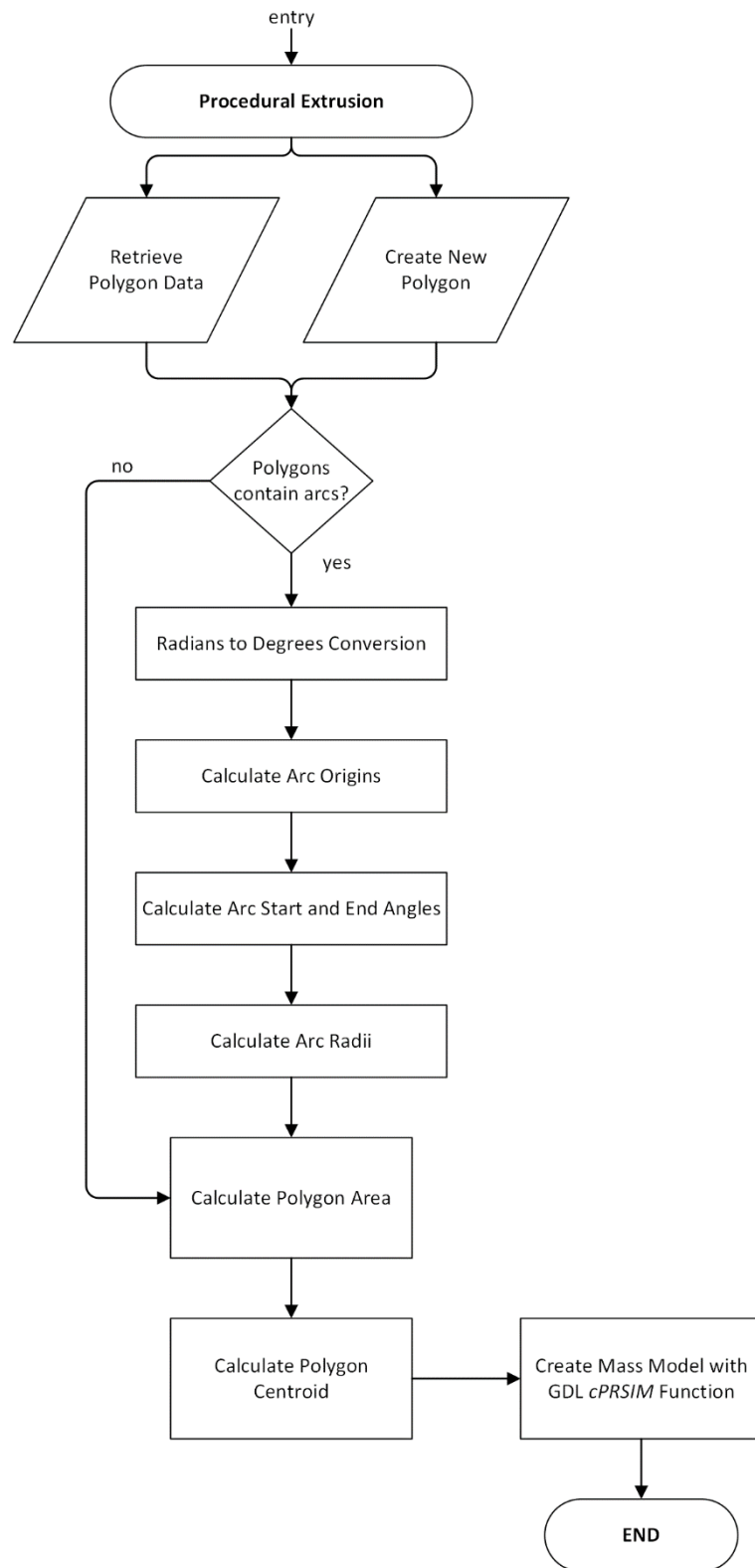


Figure 4.4: Flow diagram showing steps for rule one – Procedural Extrusion



Table 4.1: Steps for calculating arc origins using arc start and end coordinates and arc angle ( $\theta_1$ ) (Figure 4.5).

1) Calculate distance ( $L$ ) between arc start and end points using Equation 4.2.
2) Calculate angle of line between arc start and end points ( $\beta$ ) using Equation 4.3.
3) Calculate remaining internal angles ( $\theta_2$ & $\theta_3$ ) of triangle between arc origin and arc start and end points (Equation 4.4).
4) Calculate angle of line from arc start point to arc origin ( $\beta - \theta_2$ ).
5) Calculate distance from the arc start point to the arc origin using sine rule with internal triangle angles (Equation 4.5).
6) Calculate coordinates of arc origin using distance and angle from arc start point (Equation 4.6 and Equation 4.7).

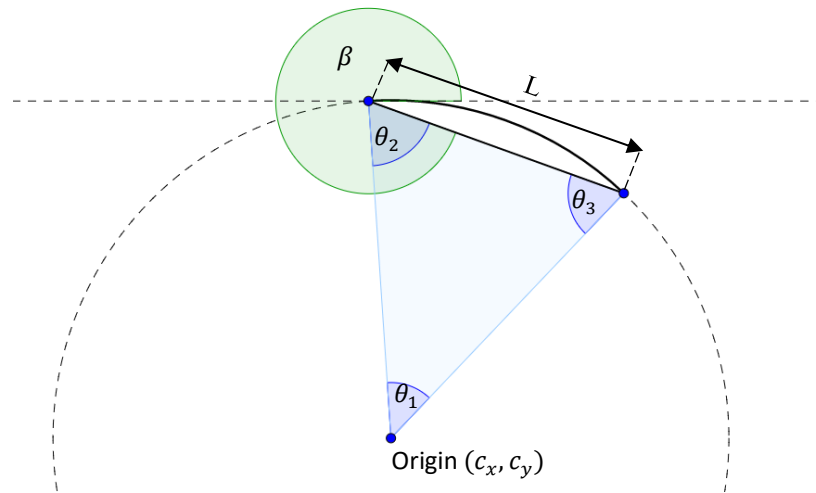


Figure 4.5: Diagram showing solution for calculating arc origins using arc start and end points and arc angle ( $\theta_1$ ).

The next step of the workflow for this procedural rule involves calculating the area of polygons (Figure 4.4). This is required to establish if a polygon has its vertices ordered in a clockwise or counter-clockwise direction. A polygon with a clockwise direction will result in a negative area while a counter-clockwise direction will result in a positive area. For consistency, all mass models are generated in a clockwise direction so the order in which polygon coordinates are used in GDL scripts is dependent on the polygon direction. The formula for calculating the area of an irregular polygon is shown

in Equation 4.8. For this formula, polygons are considered with “ $n$ ” representing the number of vertices starting at 0 and ending at “ $n - 1$ ” as the last vertex is assumed to be the same as the first for a closed polygon.

Table 4.2: Equations used in rule one, Procedural Extrusion.

$$degrees = radians \times \frac{180}{\pi}$$

Equation 4.1: Angle conversion from radians to degrees.

$$L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Equation 4.2: Formula for calculating the distance between two planar points.

$$\beta = \tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right)$$

Equation 4.3: Formula for calculating the angle of a line connecting two planar points.

$$a = \frac{(180 - \text{arc angle})}{2}$$

Equation 4.4: Formula for calculating equal angles in an Isosceles triangle.

$$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C}$$

Equation 4.5: Sine Rule

$$a^2 + b^2 = c^2$$

Equation 4.6: Pythagoras Theorem.

$$x_2 = \text{distance} \times \cos(\text{angle})$$

$$y_2 = \text{distance} \times \sin(\text{angle})$$

Equation 4.7: Formula for calculating the coordinates of an unknown point using a distance and angle from a known coordinate.

$$\text{Area} = \frac{1}{2} \left( \sum_{i=0}^{n-1} (x_i \cdot y_{i+1} - x_{i+1} \cdot y_i) \right)$$

Equation 4.8: Formula for calculating the area of an irregular polygon

$$cx = \left( \sum_{i=0}^{n-1} x_i \right) / (n - 1)$$

$$cy = \left( \sum_{i=0}^{n-1} y_i \right) / (n - 1)$$

Equation 4.9: Formula for calculating x and y coordinates of a polygon centroid.

The final calculation in the workflow for this procedural rule before generating a mass model involves computing the coordinates of the centroid of each polygon (Figure 4.4). This is also known as the centre of gravity or centre of mass. This centroid position is later used as a hotspot for interactive editing the parameter for the number of storeys in a building model. The formula for calculating this centroid position is shown in Equation 4.9. The centroid x coordinate is calculated by adding all x coordinates of vertices in a polygon and dividing by the number of vertices. Similarly, the centroid y coordinate is calculated by adding all y coordinates of vertices and dividing by the number of vertices. As with the area calculation,  $x_n$  is assumed to be the same as  $x_0$  as the polygon is closed and has the same end and start vertices. The final step of this procedural rule involves generating the 3D geometry for a mass model using the retrieved and calculated 2D polygon data (Figure 4.4).

#### **4.3.2 Rule Two: Procedural Offset**

The second rule (Figure 4.6) converts a mass model into a higher level of detail (LoD) model which is represented by walls with a uniform thickness. Walls are created from the mass model by defining an opening in the mass model and leaving just the area contained by the walls. For this rule, an offset algorithm has been developed which offsets the 2D building footprint by a distance equal to the wall thickness. This new offset polygon is used to define the opening which is removed from the original mass model. The wall thickness which is the offset distance is a parameter for this rule and can be altered by a user.

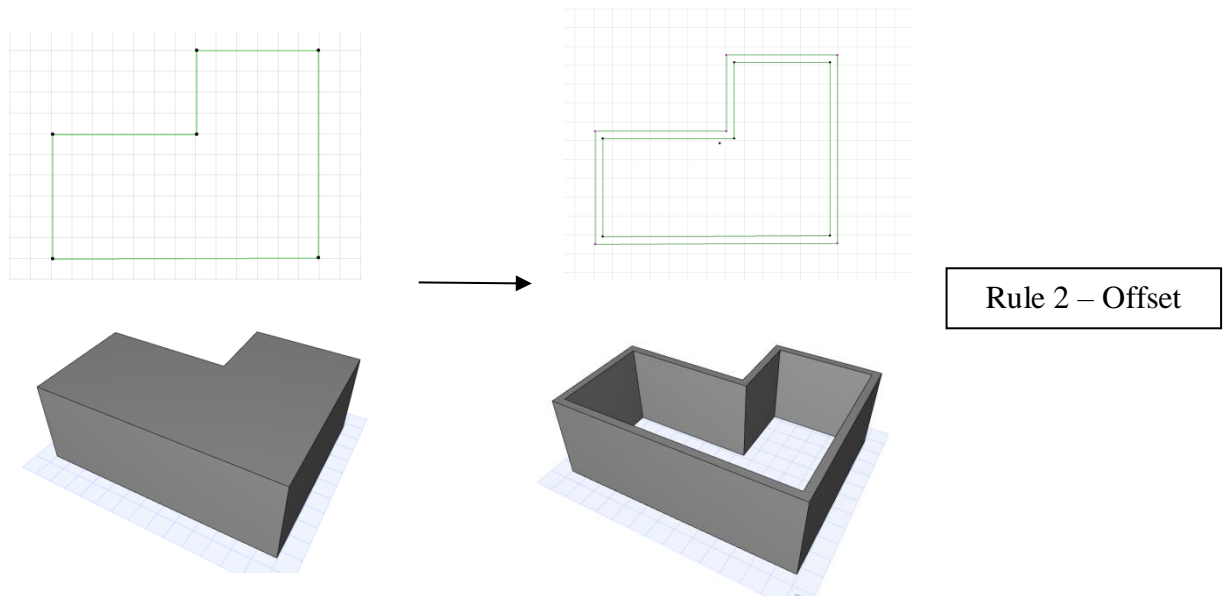


Figure 4.6: Rule Two - Offset a building footprint to convert a mass model into walls with uniform thickness

The inputs for this rule are the mass models generated with the previous rule and the accompanying polygon data (Figure 4.7). This includes polygon x and y coordinates, polygon area, arc start and end coordinates, arc angles, arc origin coordinates, arc start and end angles and arc radii. The outputs from this rule include coordinates defining a new offset polygon, coordinates for new offset arc start and end points, offset arc start and end angles, offset arc radii and new wall objects generated with a GDL *cPRISM* function. Figure 4.8 shows a workflow diagram for the steps involved in this procedural offset rule. This offset rule is applied if a user parameter for the level of detail (LoD) is set to two or more. Otherwise, the model will be displayed as a lower LoD mass model using rule one.

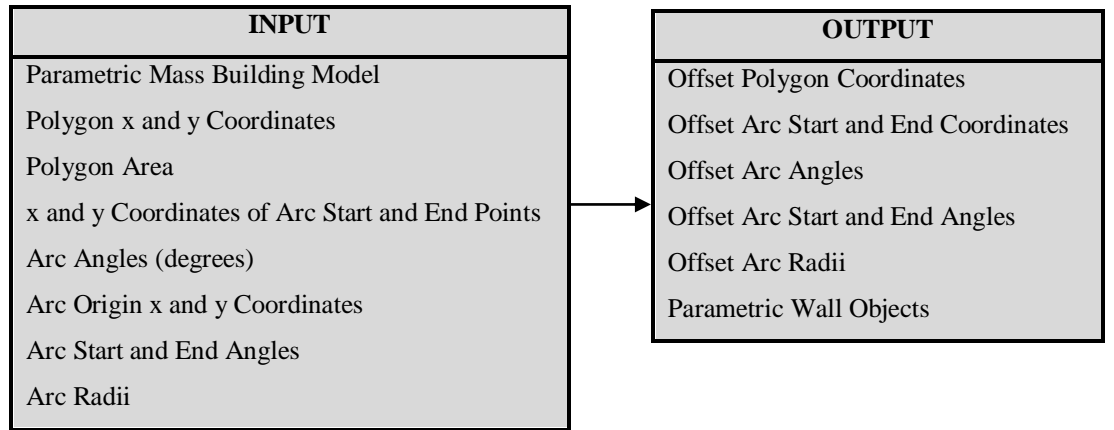


Figure 4.7: Inputs and outputs from procedural rule two – Procedural Offset

The first part of the workflow for the offset algorithm involves calculating perpendicular directions for each building side (Figure 4.8). Using the cross product formula a perpendicular direction for a line can be calculated by swapping the x and y coordinates and changing the sign of the new x coordinate (Equation 4.10).

$$\begin{pmatrix} \Delta x_{perp} \\ \Delta y_{perp} \end{pmatrix} = \begin{pmatrix} -\Delta y \\ \Delta x \end{pmatrix} = \begin{pmatrix} -(y_2 - y_1) \\ (x_2 - x_1) \end{pmatrix}$$

Equation 4.10: Formula for calculating a perpendicular direction for a line using the cross product formula.

The second stage of the workflow for the offset algorithm involves calculating start and end coordinates of each offset parallel line and arc. For offset parallel lines this is achieved using the offset distance and perpendicular direction from the original line endpoints (Equation 4.7). If a polygon contains arcs, then the offset arcs are calculated by adding or subtracting the offset distance to the original arc radius.

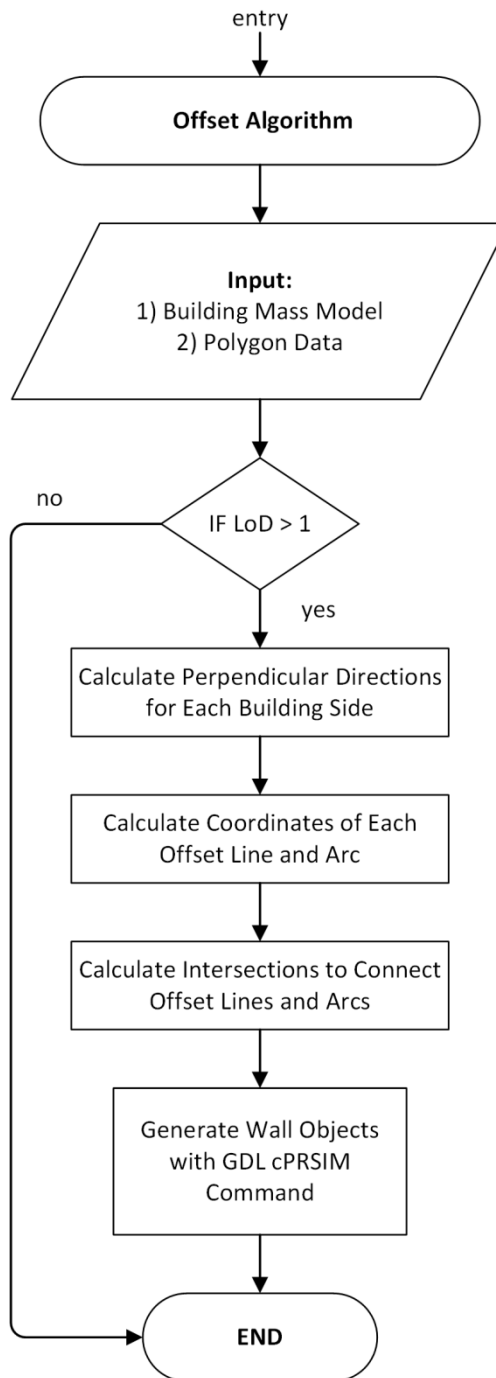


Figure 4.8: Flow diagram showing steps for rule two – Procedural Offset

As a result of being offset, the new lines and arcs are no longer connected. Individual lines and arcs are either intersecting or do not meet (Figure 4.9). To overcome this, the third stage of the offset algorithms requires the intersections between individual lines and arcs to be calculated. This involves calculating the intersection between two lines, the intersection between a line and an arc or the intersection between two arcs

depending on whether two adjacent sides are lines of arcs (Figure 4.9). Figure 4.10 shows a workflow diagram for evaluating the required intersections.

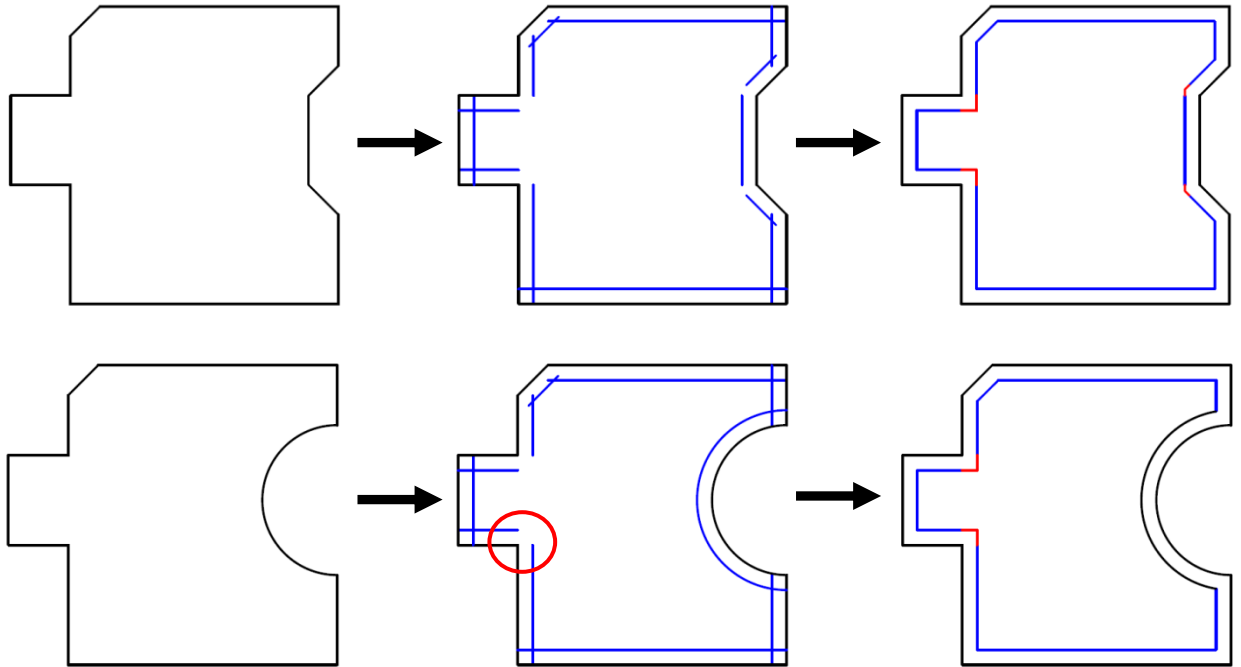


Figure 4.9: Original polygon (left), unconnected offset lines and arcs (middle) and single connected offset polygon (right)

The formula for calculating the intersection between two lines is shown in Equation 4.11. This can be used for crossing lines and lines segments that do not meet (Figure 4.11). If the denominator of this formula is zero, then the two lines are parallel and there is no unique point of intersection. This situation may arise for this offset algorithm if two adjacent polygon sides have the same direction or angle (one line divided into more than one vertex). In this case, no intersection needs to be calculated as the connection points are the original vertices between such adjacent lines.

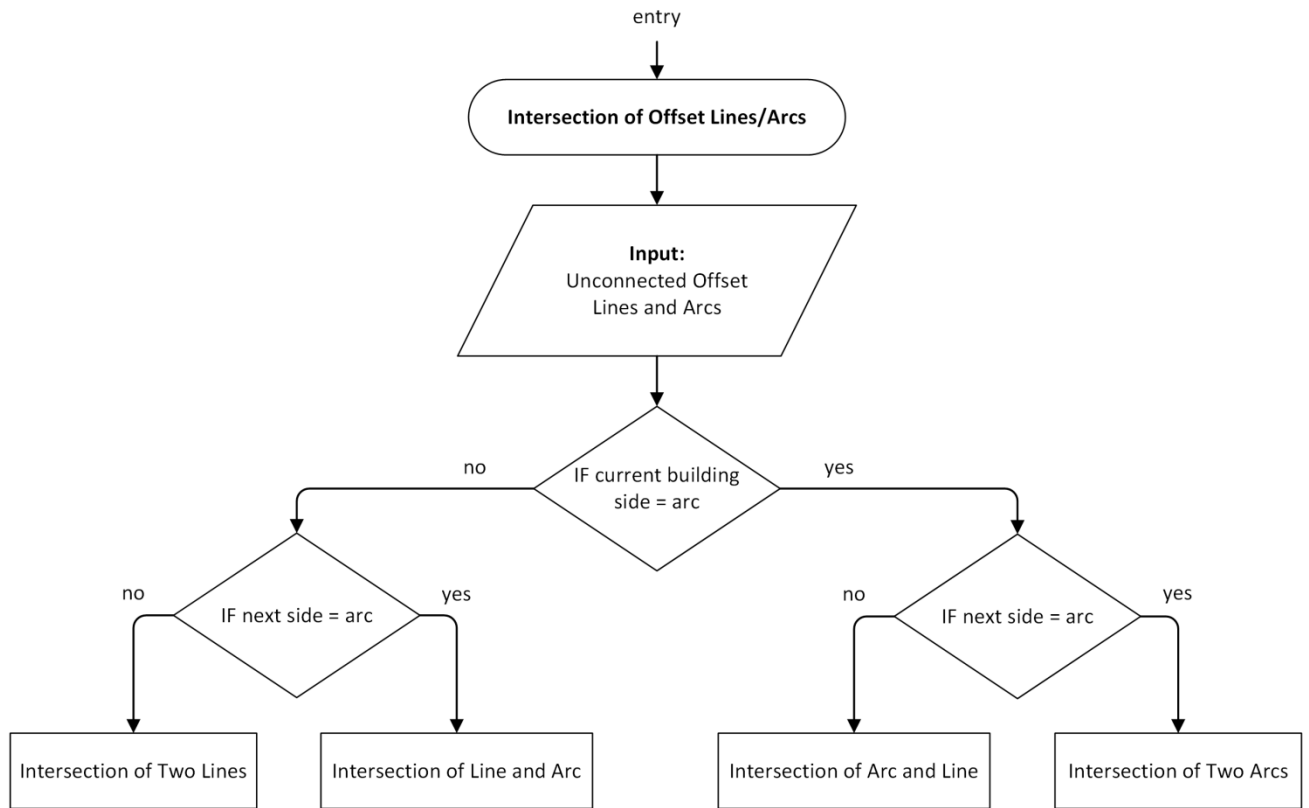


Figure 4.10: Flow diagram showing steps for calculating the intersections of offset lines and arcs.

$$P_x, P_y = \left( \frac{(x_1.y_2 - y_1.x_2)(x_3 - x_4) - (x_1 - x_2)(x_3.y_4 - y_3.x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}, \frac{(x_1.y_2 - y_1.x_2)(y_3 - y_4) - (y_1 - y_2)(x_3.y_4 - y_3.x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \right)$$

Equation 4.11: Formula for calculating the intersection of two lines given two points on each line

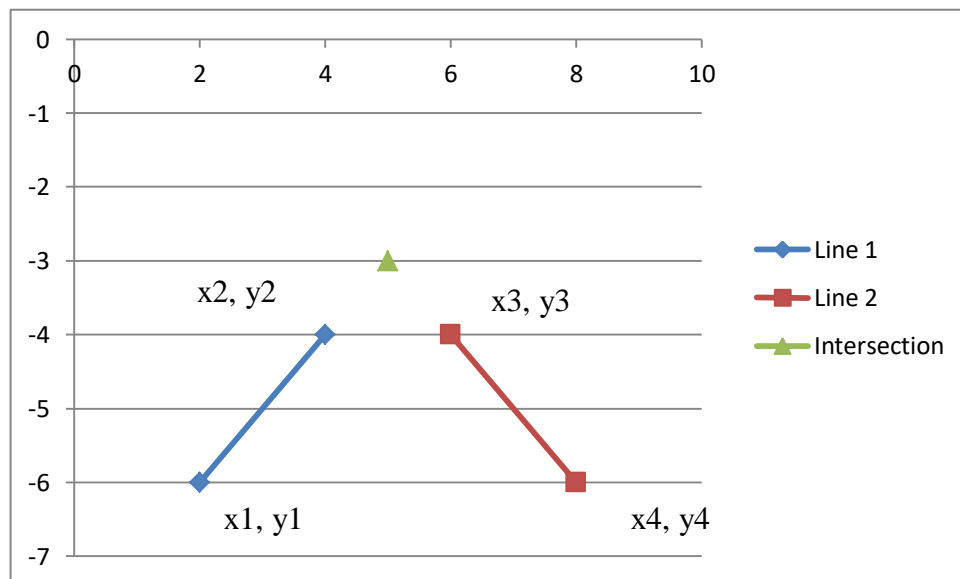


Figure 4.11: Intersection of two line using Equation 4.11.



The steps for calculating the intersection of a line and arc are shown in Table 4.3 and Equation 4.12 to Equation 4.16 (Watson, 2009). Intersection points are solved by connecting a triangle between the arc origin and the interesting line as shown in Figure 4.12. Side  $h$  of this triangle is perpendicular to the line which intersects the arc. By solving the sides  $w$  and  $h$  of this triangle, the points of intersection are found.

Table 4.3: Steps for calculating the intersection of a line and arc (Figure 4.12).

1) Calculate distance of line (Equation 4.2).
2) Calculate a unit direction vector for line (Equation 4.12).
3) Calculate perpendicular direction for line (Equation 4.13).
4) Calculate perpendicular distance ( $h$ ) from line to arc origin (Equation 4.14).
5) Calculate remaining side of triangle ( $w$ ) using Pythagoras Theorem (Equation 4.15).
6) Calculate two intersection points using distances and directions from the known origin coordinates ( $x_c, y_c$ ) (Equation 4.16).
7) Test if intersection points lie on line and arc segments by checking limits of arc and line segments.

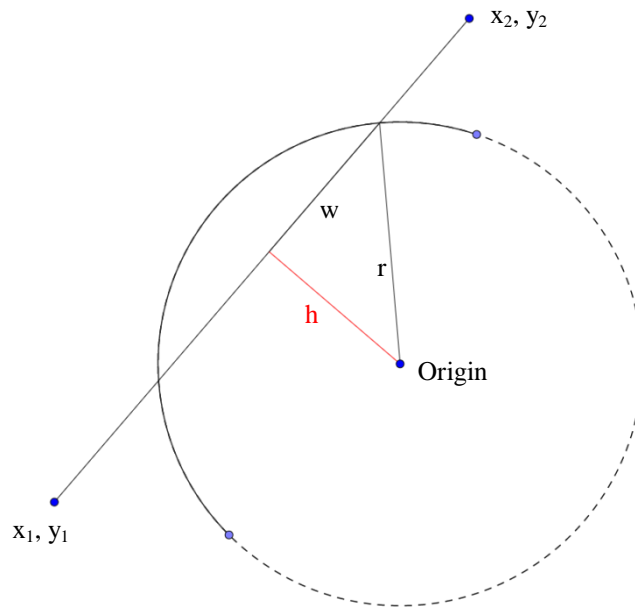


Figure 4.12: Diagram showing solution for calculating the intersection of a line and arc (Table 4.3).

Table 4.4: Equations used to calculate the intersection points between a line and arc.

$$\hat{u} = \begin{pmatrix} (x_2 - x_1)/L \\ (y_2 - y_1)/L \end{pmatrix}$$

Equation 4.12: Formula for calculating a unit direction vector  $\hat{u}$  for a line where L is the total distance of the line.

$$\begin{pmatrix} \hat{v}_x \\ \hat{v}_y \end{pmatrix} = \begin{pmatrix} -\hat{u}_x \\ \hat{u}_y \end{pmatrix}$$

Equation 4.13: Formula for calculating a unit vector  $\hat{v}$  which is perpendicular to the unit direction vector  $\hat{u}$ .

$$h = \underline{p} \cdot \hat{v} = \begin{pmatrix} x_1 - x_c \\ y_1 - y_c \end{pmatrix} \cdot \begin{pmatrix} -u_y \\ u_x \end{pmatrix}$$

$$h = u_x(y_1 - y_c) - u_y(x_1 - x_c)$$

Equation 4.14: Dot product formula used for calculating perpendicular distance (h) in Figure 4.12. Vector  $\underline{p}$  is the vector starting at the arc origin ( $x_c, y_c$ ) and ending at line endpoint ( $x_1, y_1$ ).

$$w = \sqrt{r^2 - h^2}$$

Equation 4.15: Pythagoras Theorem used to calculate distance (w) in Figure 4.12.

$$\begin{pmatrix} x_L \\ y_L \end{pmatrix} = \begin{pmatrix} x_c \\ y_c \end{pmatrix} + h \begin{pmatrix} v_x \\ v_y \end{pmatrix} - w \begin{pmatrix} u_x \\ u_y \end{pmatrix}$$

$$\begin{pmatrix} x_R \\ y_R \end{pmatrix} = \begin{pmatrix} x_c \\ y_c \end{pmatrix} + h \begin{pmatrix} v_x \\ v_y \end{pmatrix} + w \begin{pmatrix} u_x \\ u_y \end{pmatrix}$$

Equation 4.16: Formula for calculating two intersection points (left (L) and right (R)) using distances and directions from arc origin ( $x_c, y_c$ ).

If two adjacent sides of a polygon are arcs then an intersection between two arcs is calculated. Table 4.5 shows the steps for calculating the intersection of two arcs using Equation 4.17 to Equation 4.19 (Watson, 2009). Intersection points are solved by connecting triangles between both arc origins and intersection points (Figure 4.13). The cosine rule is then used to solve for angles  $\theta_1$  and  $\theta_2$ , which are used to calculate the intersection points.

Table 4.5: Steps for calculating the intersection of two arcs (Figure 4.13).

1) Test if arcs are intersecting (Equation 4.17).
2) Calculate angles $\Theta_1$ and $\Theta_2$ in Figure 4.13 using the cosine rule (Equation 4.18).
3) Calculate a unit direction vector $\hat{u}$ from $(x_{c1}, y_{c1})$ to $(x_{c2}, y_{c2})$ using Equation 4.12.
4) Calculate a perpendicular direction vector $\underline{v}$ which is perpendicular to $\hat{u}$ (Equation 4.13).
5) Calculate intersection points by expressing the intersection points as components of $\hat{u}$ and $\underline{v}$ (Equation 4.19).
6) Test if intersections fall within the arc segments by comparing the intersection points and the limits of each arc segments.

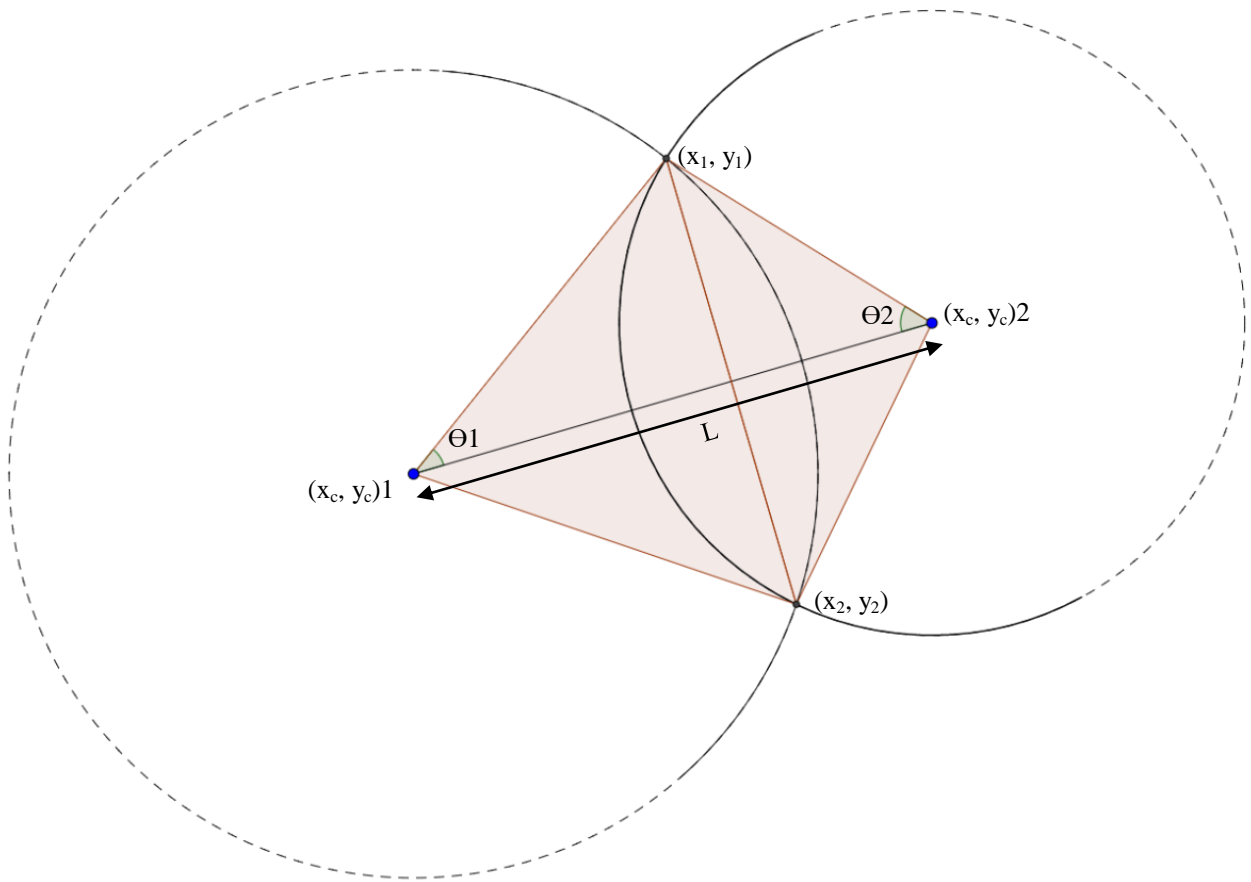


Figure 4.13: Diagram showing solution for calculating the intersection of two arcs (Table 4.5).

Table 4.6: Equations used to calculate the intersection points between two arcs

$$\sqrt{(x_{c1} - x_{c2})^2 + (y_{c1} - y_{c2})^2} \leq (R_1 + R_2)$$

Equation 4.17: Formula to test if two arcs are intersecting by checking if the distance between the arc centres is less than the sum of the radii.

$$\theta_1 = \cos^{-1} \left( \frac{R_1^2 + L^2 - R_2^2}{2R_2L} \right)$$

$$\theta_2 = \cos^{-1} \left( \frac{R_2^2 + L^2 - R_1^2}{2R_2L} \right)$$

Equation 4.18: Cosine Rule for calculating angles  $\theta_1$  and  $\theta_2$  in Figure 4.13.

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_{c1} \\ y_{c1} \end{pmatrix} + R_1 \cos \theta_1 \begin{pmatrix} u_x \\ u_y \end{pmatrix} \pm R_1 \sin \theta_1 \begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

Equation 4.19: Formula for calculating intersection points between two arcs by expressing the intersection points as components of unit direction vector  $\hat{u}$  and perpendicular vector  $\underline{v}$  (Figure 4.13).

The final stage of the offset algorithm involves using the coordinates of the offset polygon to define a hole in the previous mass model (Figure 4.8). This enables the mass model to be represented in a higher level of detail model with wall components.

### 4.3.3 Rule Three: Repeat

The third procedural rule is a repeat rule that is used to repeat a wall instance for any number of floors specified by a user parameter (Figure 4.14). A loop is used to create new wall instances for each floor. By default new wall instances created have the same footprint as the ground floor but using rule one for extruding a building footprint it is also possible to create buildings with different footprints on each floor. When rule one (Procedural Extrusion) and rule three (Repeat) are applied the heights of each floor (extrusion height) are automatically calculated by applying the classical proportions and architectural rules shown in Figure 3.3. The inputs for this rule are the parametric wall objects created with previous rules, associated polygon data and architectural proportions which are used to setup initial floor heights with classical proportions

(Figure 4.15). The outputs from this rule include parametric wall objects for any number of floors with classically proportioned floor heights, in addition to capabilities for interactive editing the number of floors and floor heights (Figure 4.15).

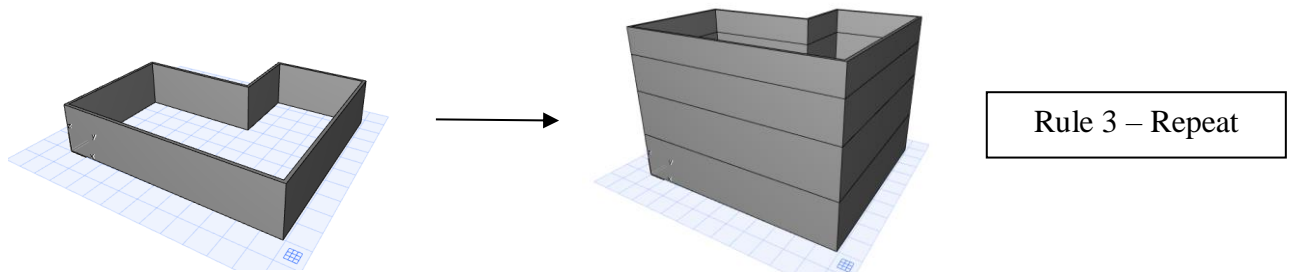


Figure 4.14: Rule Three – Repeat instances of a wall for any number of floors as set by a user.

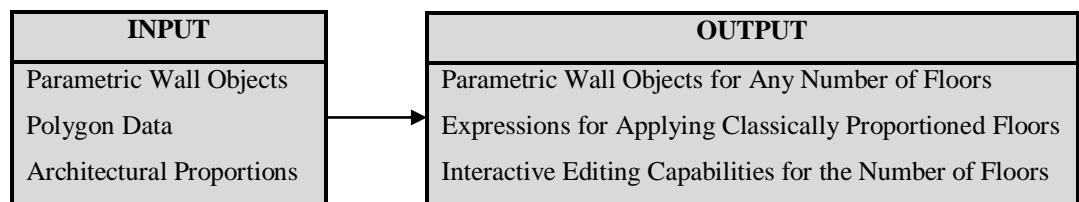


Figure 4.15: Inputs and outputs from procedural rule three – Repeat Rule

Figure 4.16 shows a workflow diagram for the steps involved in this procedural rule. The first step involves calculating and applying classical proportions to determine the initial heights for each floor (Figure 4.16). Classical proportions outlined in Figure 3.3 and Figure 3.5 are used to calculate the initial floor heights. These proportions are calculated with a series of parametric expressions where floor heights are expressed in relation to window heights and positions (Figure 3.5). After the initial floor heights are calculated and applied, the second stage of the workflow repeats the previously generated wall geometry for any number of floors.

The final part of the workflow for this rule is a search algorithm to enable interactive editing for the number of floors. Interactive editing works by clicking and dragging hotspots on a model to change a parameter. This works well for dimension type

parameters which are based on a linear distance from a defined base point. A parameter for the number of floors, however, is not a dimension parameter but an integer which cannot be interactively edited in the same way. To overcome this, a different parameter for the building height is instead edited interactively by a hotspot in the 3D window (Figure 4.17). The building height set by a user is then used to determine the number of floors. The building height set by a user is compared to the set of pre-determined floor heights. Each floor height added to the sum of the previous floor heights is compared to the new building height. When the correct floor is found based on the building height the new required geometry for that number of floors is generated. Figure 4.18 shows a flow diagram for this final stage of the repeat rule.

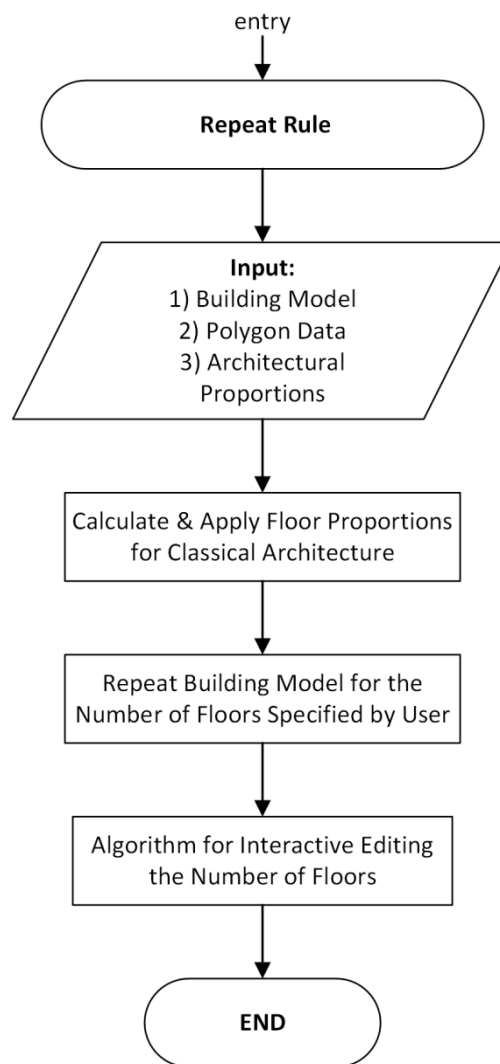


Figure 4.16: Flow diagram showing steps for rule three – Repeat Rule.

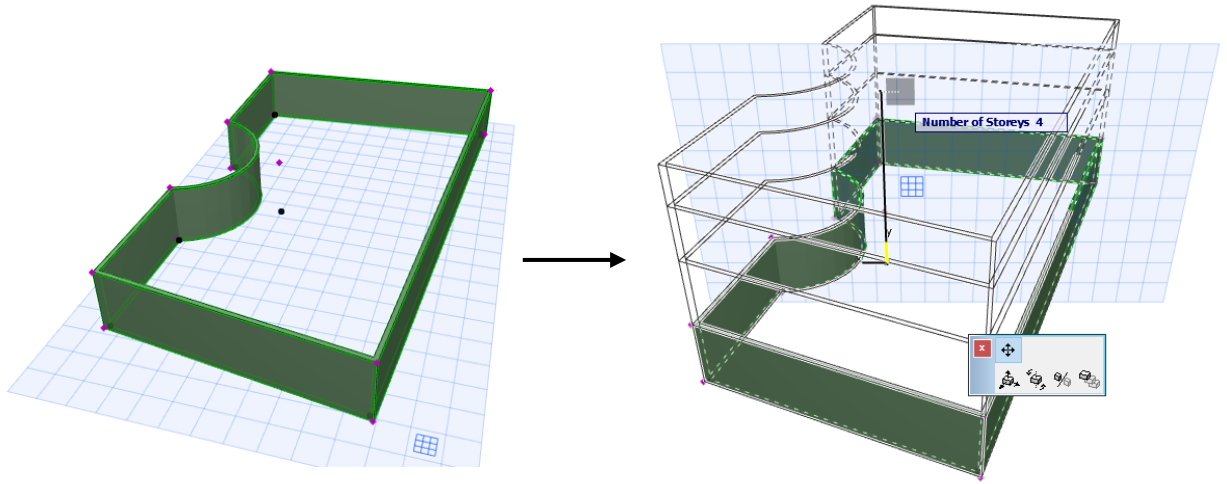


Figure 4.17: Interactive editing the number of floors by clicking and dragging a hotspot.

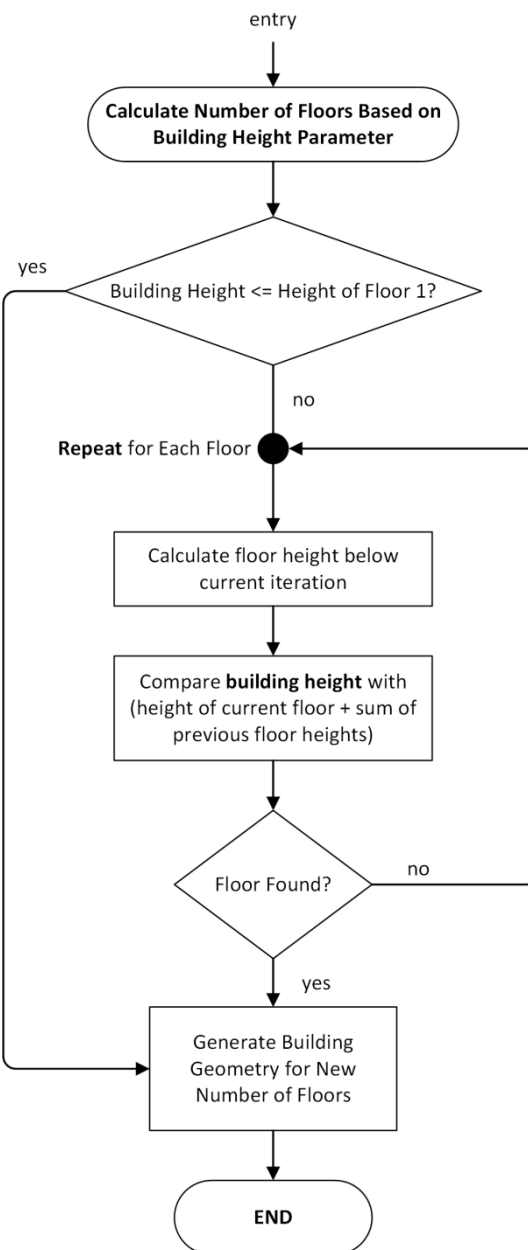


Figure 4.18: Flow diagram showing steps for calculating the number of floors after interactive editing.

#### 4.3.4 Rule Four: Split Rule

The fourth procedural rule is a split rule which is used to subdivide a façade or floor into any number of tiles which may contain openings (Figure 4.19). Unlike the previous procedural façade prototype where tiles are created by repeating tile instances, the procedural building prototype creates tile instances by subdividing existing wall geometry. Users can apply a split rule to a complete building, all floors on a particular building side or a specific floor on a building side. Different floors on a façade can also have different numbers of tiles. The number of tiles to be created is a parameter for this rule.

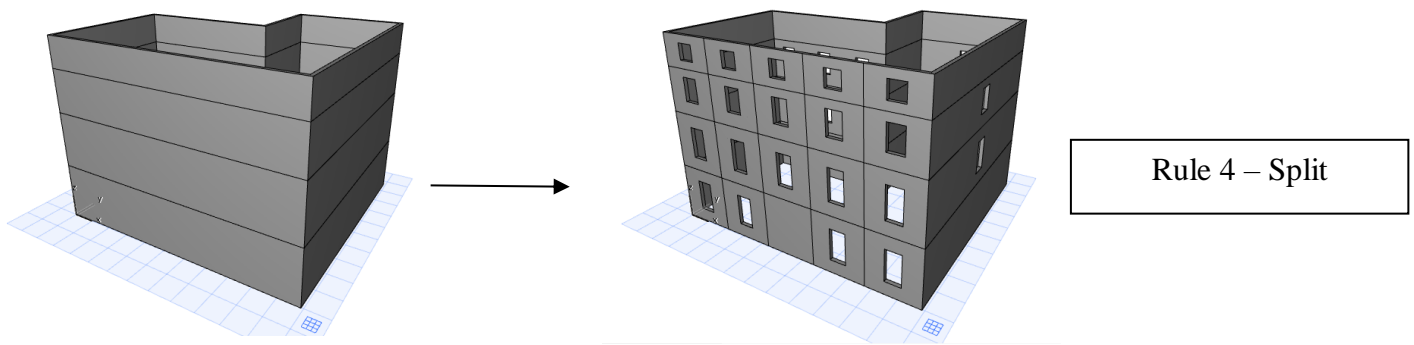


Figure 4.19: Rule Four - Split all floors or a specific floor on a building side into any number of tiles set by a user

The inputs for this rule are the parametric wall objects created with previous rules, associated polygon data and architectural data which is used to apply classical proportions to window openings (Figure 4.20). The outputs from this rule include coordinates of split lines which subdivide a planar or curved façade or floor, coordinates for tiles and openings on a subdivided planar or curved façade or floor, expressions for calculating and applying classical proportions for window openings and parametric building model geometry which contains walls and openings (Figure 4.20).



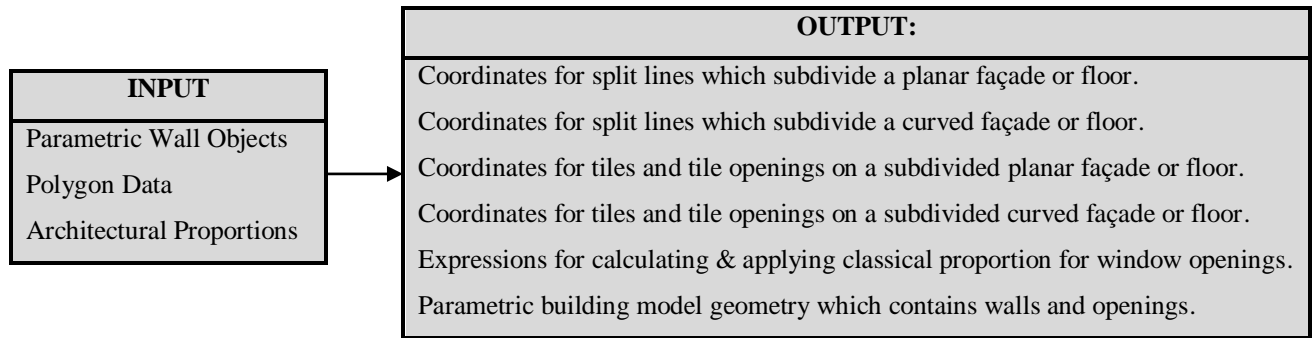


Figure 4.20: Inputs and outputs from procedural rule four – Split Rule

Figure 4.21 shows a workflow diagram for the steps involved in this procedural rule. When a split rule is applied the first step involves calculating the coordinates of each split line which divides a façade or floor into smaller tiles. These split line coordinates are used to populate tile lengths and tile coordinates. For a planar building side, split line coordinates are calculated by dividing the delta x and delta y of facade endpoints equally by the number of tiles as shown in Figure 4.22 and Equation 4.20. The steps for splitting a curved building face are shown in Figure 4.23 and Equation 4.21 and Equation 4.22. For a curved building façade, an arc angle is divided by the number of tiles. This relative arc angle is then used to calculate absolute angles of lines from an arc origin to split line points on the arc (Equation 4.21). The coordinates of split lines on an arc are then calculated using the split line angles and the arc radius from known arc origin coordinates (Equation 4.22).

Next, the tile lengths are calculated and stored using the split line coordinates. The distance formula in Equation 4.2 is used to calculate the tile lengths between split lines. For curved building faces, a linear tile length distance is also calculated between split lines and stored (Figure 4.23). These linear tile lengths for curved building faces are later used to calculate window widths on curved surfaces.

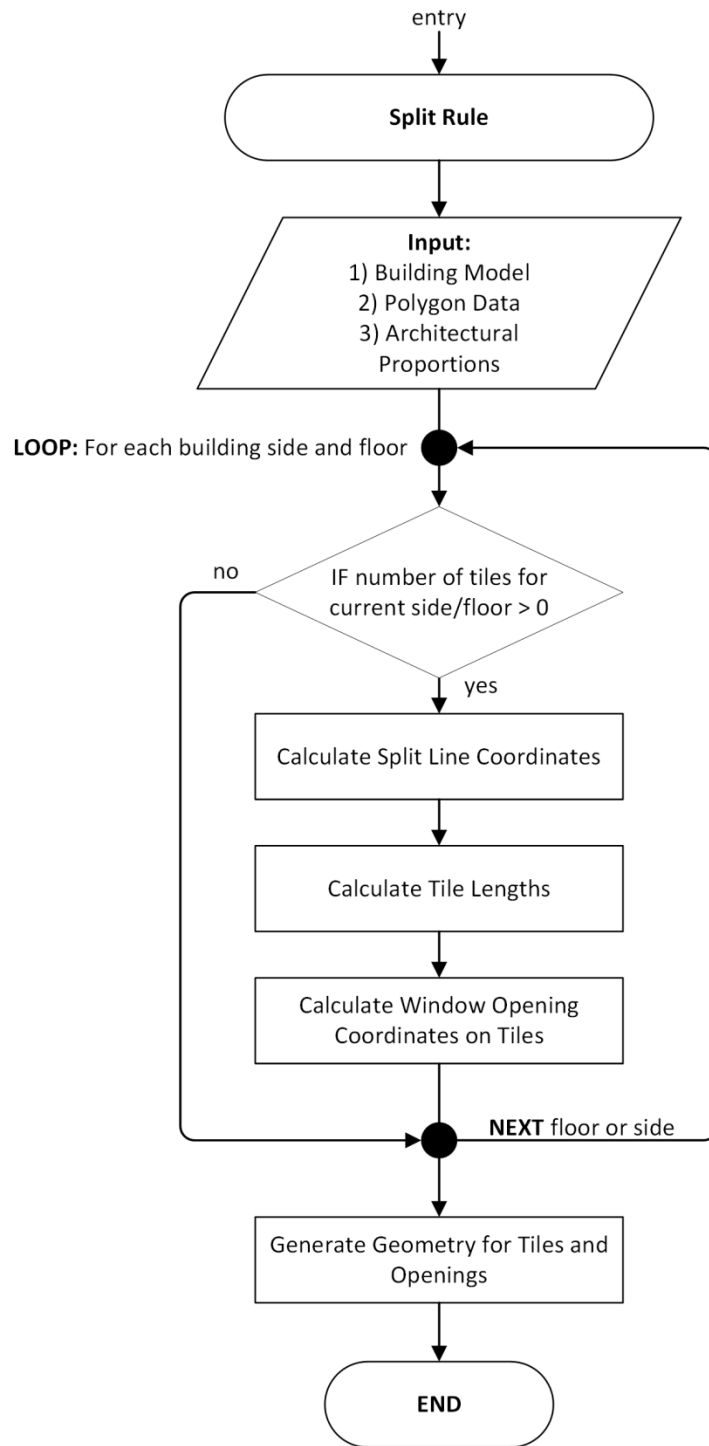


Figure 4.21: Flow diagram showing steps for rule four – Split Rule.

The third stage in this procedural rule involves calculating window opening coordinates on tiles. Initial window sizes and positions are calculated using classical proportions and rules shown in Figure 3.5. Depending on the position of a tile in a building, different opening sizes and positions are applied.

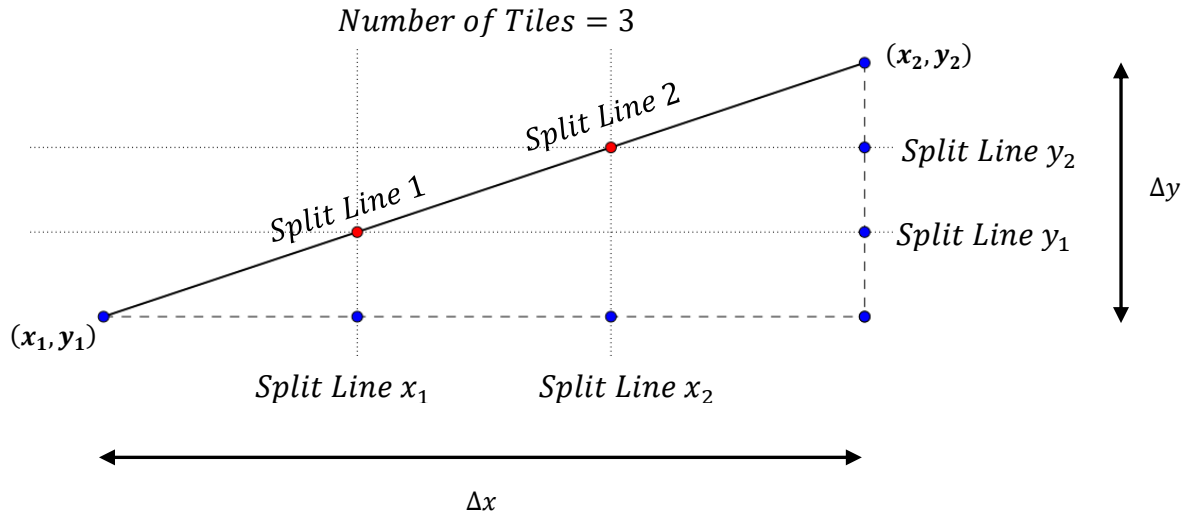


Figure 4.22: Diagram showing solution for calculating split line coordinates for subdividing a planar building façade (Equation 4.20).

$$\text{Split Line } x_{\text{lineNumber}} = x_1 + \left( \left( \frac{x_2 - x_1}{\text{Number of Tiles}} \right) * \text{lineNumber} \right)$$

$$\text{Split Line } y_{\text{lineNumber}} = y_1 + \left( \left( \frac{y_2 - y_1}{\text{Number of Tiles}} \right) * \text{lineNumber} \right)$$

Equation 4.20: Formula to calculate split line x and y coordinates for subdividing a planar building façade where (x<sub>1</sub>, y<sub>1</sub>) and (x<sub>2</sub>, y<sub>2</sub>) represent façade endpoints.

For a planar building side, tile openings are defined with coordinates relative to a local origin at the lower left-hand corner of a tile (Figure 4.24). X and Y axes of this coordinate system are rotated to be aligned to the plane of the tile (Figure 4.24). This allows openings with different sizes and positions to be defined in the same coordinate system and later aligned to any building façade using coordinate transformations. Tile corners and opening positions are defined as parameters. A set of parametric expressions define the initial coordinates for tiles and openings based on the classical proportions in Figure 3.5.

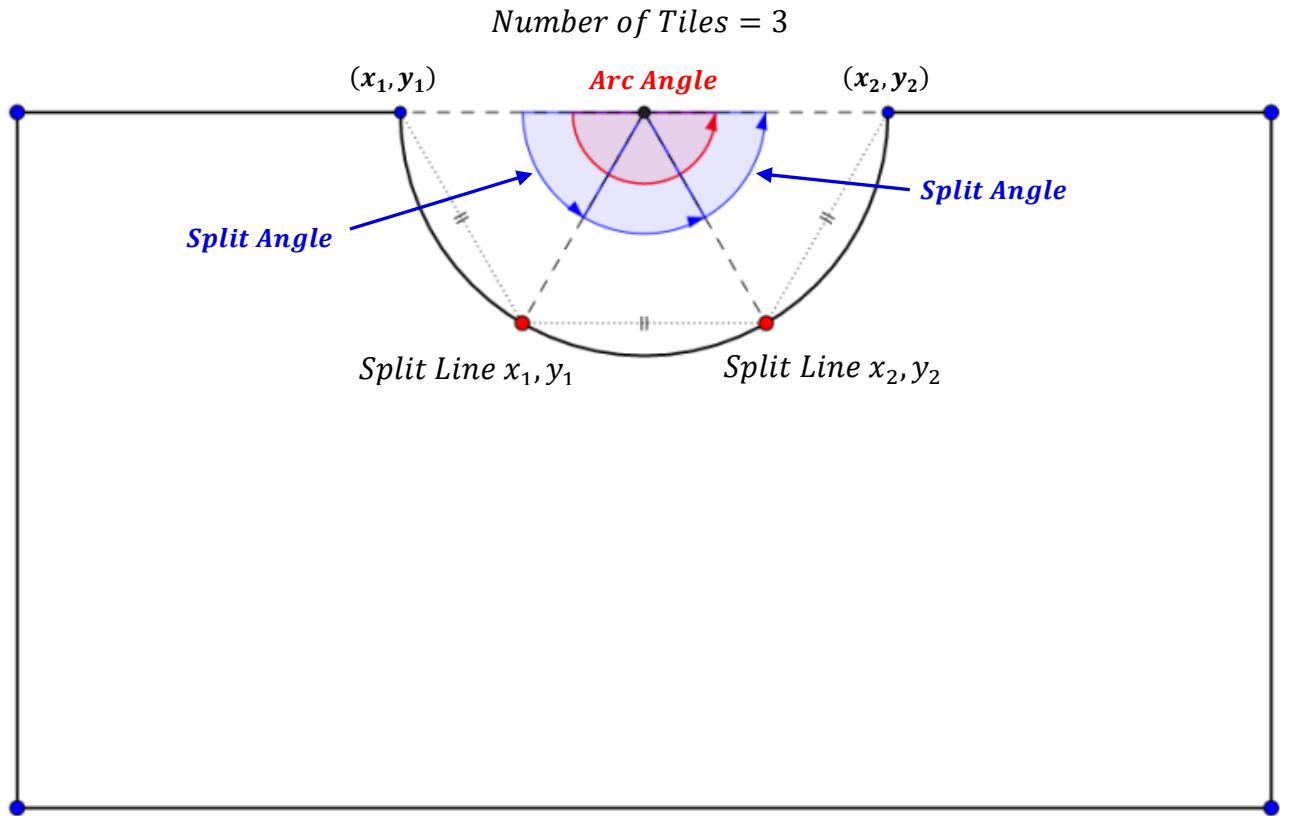


Figure 4.23: Diagram showing solution for calculating split line coordinates for subdividing a curved building face.

Table 4.7: Equations for subdividing a curved building face.

$$Split\ Angle_{lineNumber} = arc\ start\ angle + \left( \left( \frac{arc\ angle}{Number\ of\ Tiles} \right) * lineNumber \right)$$

Equation 4.21: Formula to calculate split line angles from arc origin to each split line for subdividing a curved building face.

$$Split\ Line\ x_{lineNumber} = arc\ origin_x + (arc\ radius * \cos(Split\ Angle_{lineNumber}))$$

$$Split\ Line\ y_{lineNumber} = arc\ origin_y + (arc\ radius * \sin(Split\ Angle_{lineNumber}))$$

Equation 4.22: Formula to calculate split line coordinates for subdividing a curved building face.

Window openings on a curved building face are defined differently to planar openings.

Instead of window openings being defined relative to a local origin, window openings

on curved building faces are defined using absolute coordinates where  $x$  and  $y$  coordinates represent the opening positions in plan.

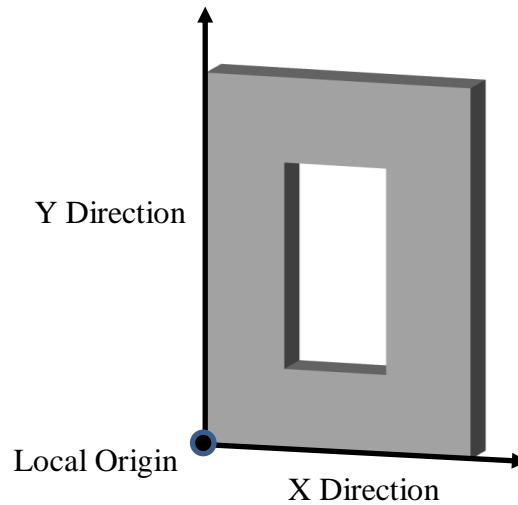


Figure 4.24: Wall tile (*TW*) containing opening showing local origin and relative coordinate systems aligned to the plane of tile.

The steps for calculating window opening positions in plan on curved building faces are shown in Table 4.8 and Figure 4.25 to Figure 4.29. Window positions are first calculated on linear tiles connecting split lines and then projected onto curved wall surfaces.

Table 4.8: Steps for calculating window openings on curved building faces.

1)	Calculate distances to window points on planar tiles (Figure 4.25) using classical proportions.
2)	Calculate absolute coordinates of window opening points on planar tiles using Equation 4.7 (Figure 4.26). For this equation, the distances to windows points on planar tiles and angles (tile angles) are used from known points (split line points) to calculate the coordinates.
3)	Calculate coordinates of perpendicular lines from window openings on planar tiles using Equation 4.10 (Figure 4.26).
4)	Calculate intersections of perpendicular lines and curved wall face (arc) using the steps in Table 4.3 (Figure 4.27).
5)	Calculate the remaining two points on window openings by extending lines from the arc origin through the first two opening points on an arc. Equation 4.3 is used to calculate the angle of these lines while Equation 4.7 is used to calculate the remaining opening coordinates. For Equation 4.7 the radius of the offset arc is used as a distance and the line angles are used from a known arc origin point to calculate the unknown coordinates.

Opening widths on curved wall faces are not uniform as they follow the shape of the arc that defines the curve. The window widths calculated from classical proportions are

maintained and are used as the minimum window width on a curved wall surface. The final window positions calculated with the steps in Table 4.8 are shown in Figure 4.29.

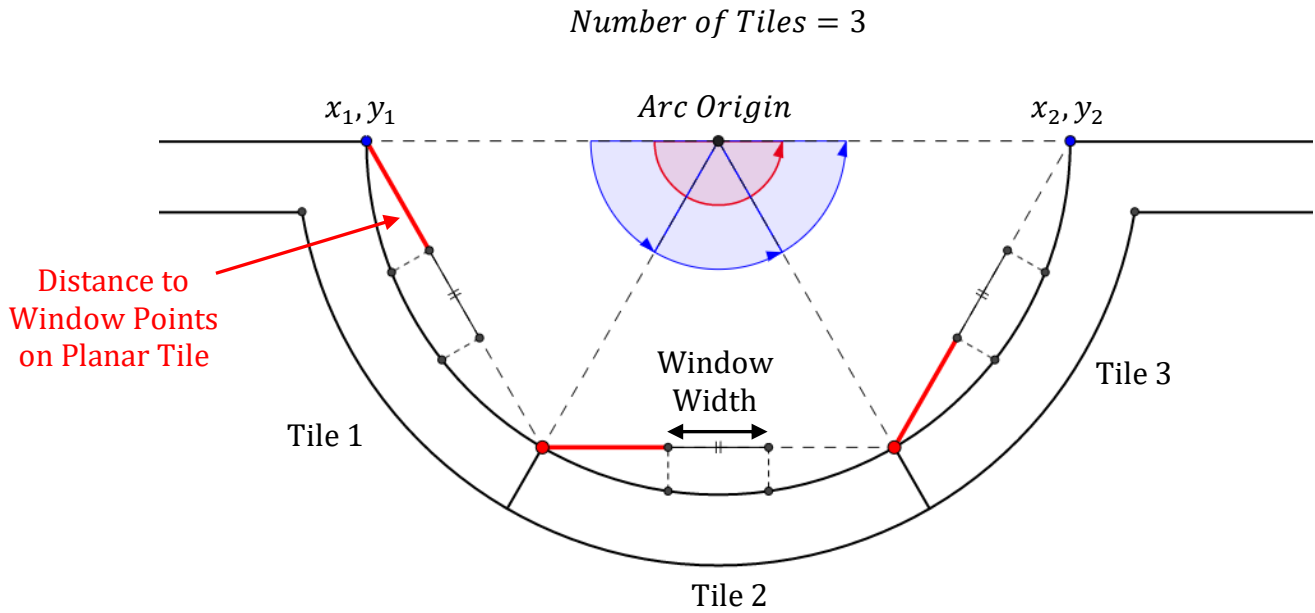


Figure 4.25: Diagram for step one in Table 4.8 to calculate distance to window points on planar tiles (lines shown in red).

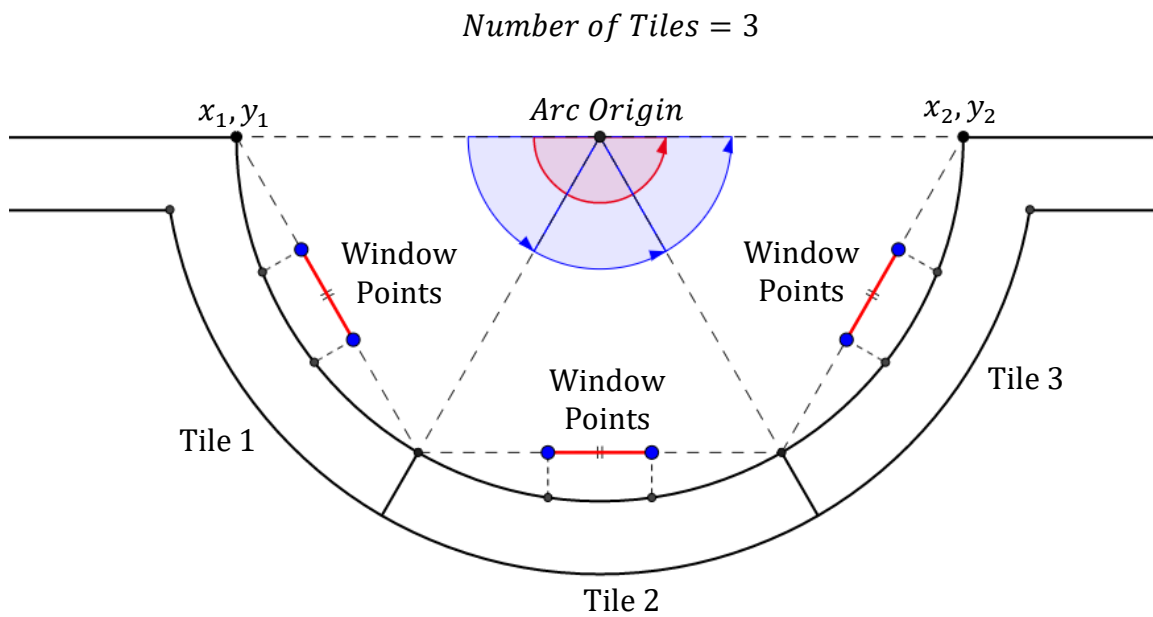


Figure 4.26: Diagram for step two in Table 4.8 to calculate absolute coordinates of window points on linear tiles (points shown in blue).

*Number of Tiles = 3*

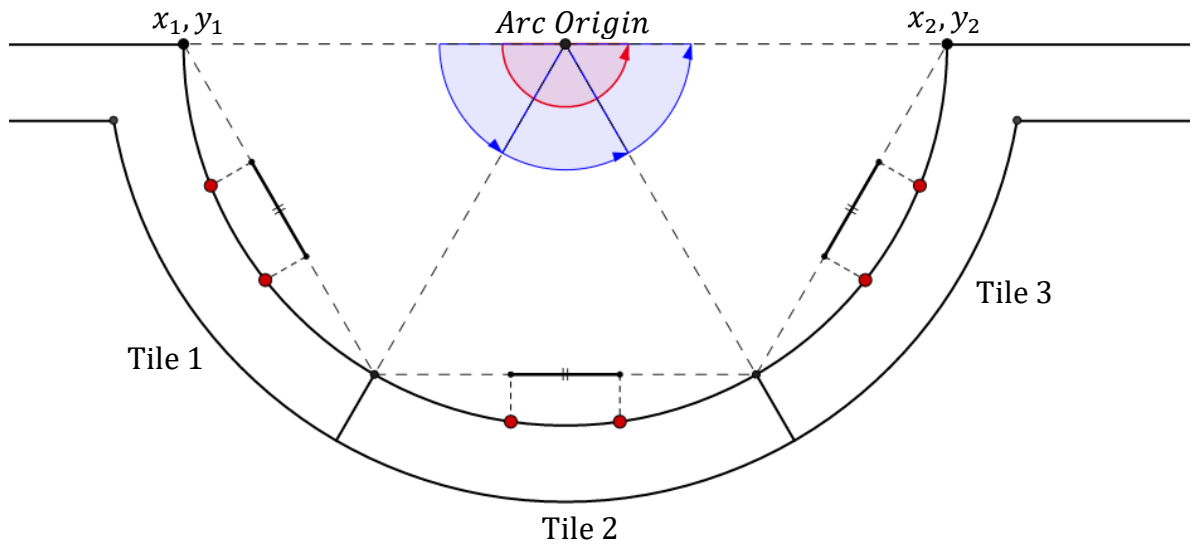


Figure 4.27: Diagram for step four in Table 4.8 to calculate intersection points shown in red.

*Number of Tiles = 3*

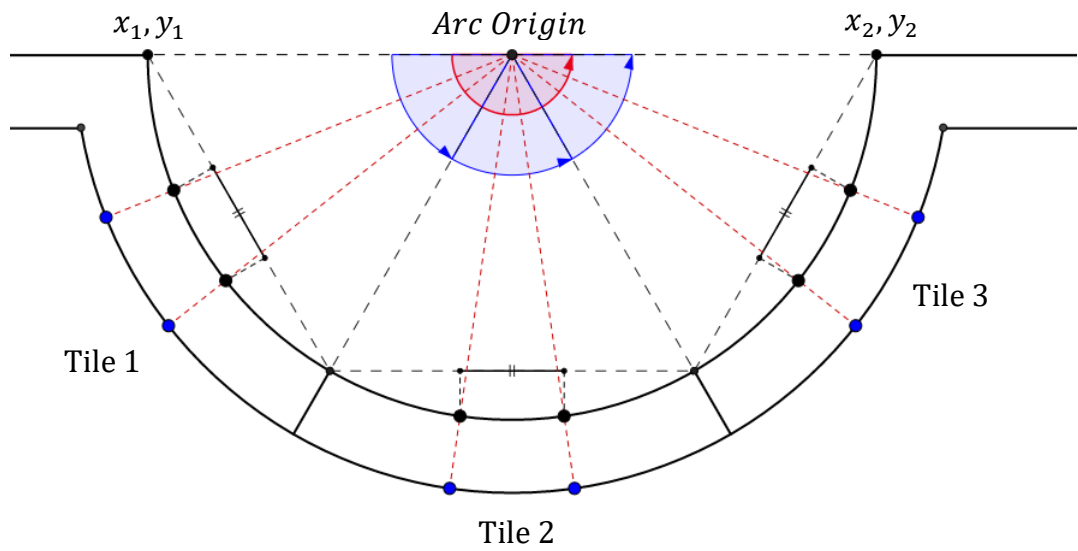


Figure 4.28: Diagram for step five in Table 4.8 to calculate remaining window opening points shown with blue points.

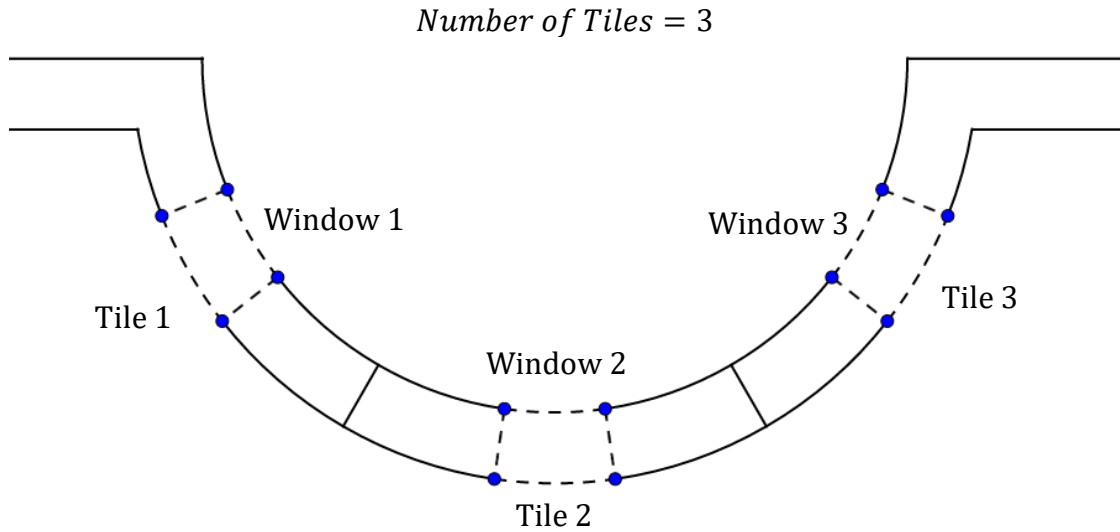


Figure 4.29: Final window plan positions calculated after applying steps in Table 4.8.

Z coordinates for window openings on curved wall faces are next calculated using formation levels and window heights as defined by classical proportions (Figure 3.5). A set of parametric expressions are again used to define these proportions as parameters. After all opening and tile coordinates are calculated, the final stage of the split rule involves generating the geometry for tiles and openings (Figure 4.21). Tiles lengths are shown graphically on a building by placing 3D lines using the split line coordinates (Figure 4.30). Boolean operations are used with opening coordinates to cut window openings from wall geometry. Figure 4.30 shows an example of the procedural split rule applied to curved and planar building faces. Openings and tiles are generated initially with classical proportions. However, once created users can graphically edit the tile distances and openings to accurately map geometry to specific survey data.



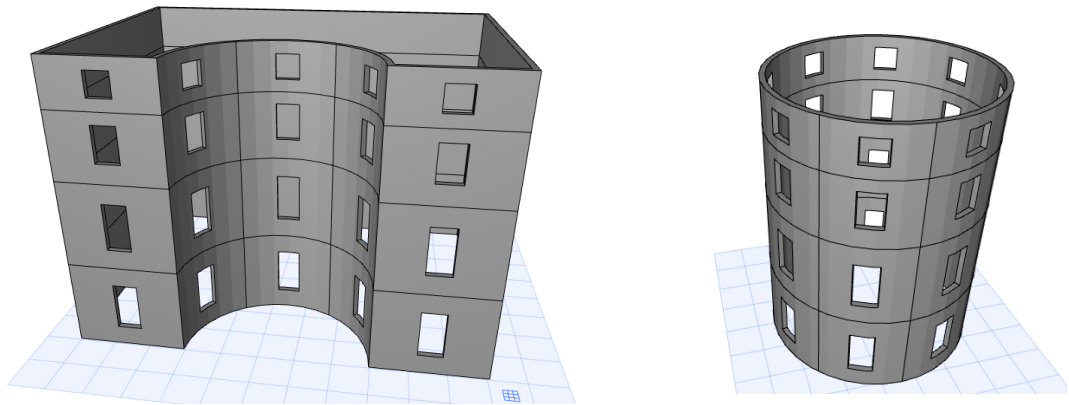


Figure 4.30: Procedural rules applied to curved geometry.

When a tile is created with a split rule, attributes describing the semantic position of that tile on the building are also stored with the tile. These attributes include the tile number on a particular floor, the floor number and the building side number (Figure 4.31). By dividing a building into tiles referenced with their semantic position it allows any part of the building to be easily and quickly identified and accessed by the program. This also allows changes to be applied to semantic groups such as editing a group of objects that are referenced to the particular floor or building side.

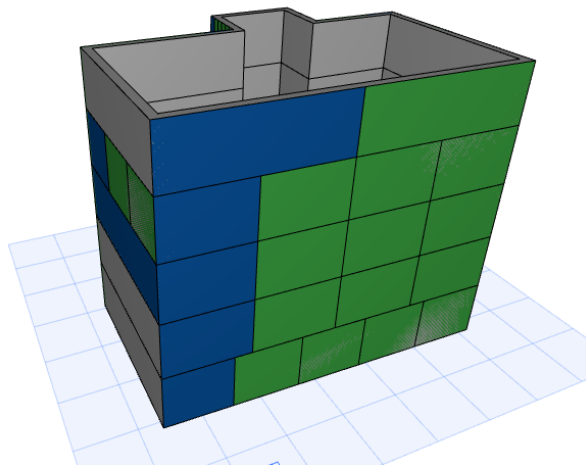


Figure 4.31: Tiles referenced by their semantic position on a building. Tiles are numbered left to right on each floor. In this image, the first tile on each floor is highlighted in blue and additional tiles are highlighted in green.

#### 4.3.5 Rule Five: Replacement Rule

Once a building structure has been created with rules one to four, then additional parametric library objects and shapes can be automatically added to the building. The fifth rule, a replacement rule, is used to replace existing geometry with new shapes or add new shapes to existing geometry. The referencing and semantic information attached to tiles allows additional detail and objects to be easily placed anywhere on the building. Objects can be added to a specific tile or groups of tiles with semantic attributes. When an object is placed on a tile the semantic attributes describing the position of the tile are also associated with the instance of the object placed on that tile. An example of an application of the replacement rule can be seen in Figure 4.32 where a wall tile ( $TW$ ) is replaced by a wall tile containing and parametric sash window ( $TW + W$ ).

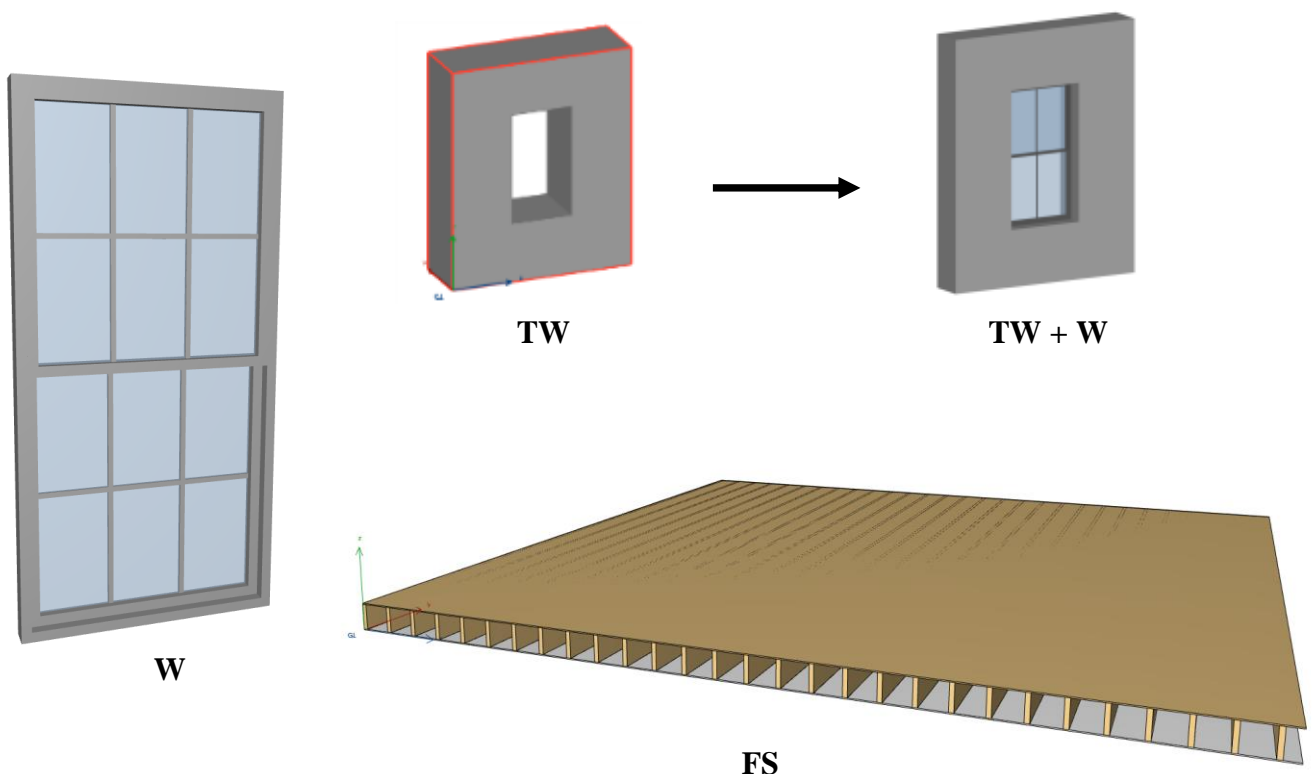


Figure 4.32: Rule Five – Replacement rule to add new parametric library objects or shapes and replace existing geometry.

When adding new objects and shapes with the replacement rule, the positions are not limited to tiles but can be placed anywhere on a building. A parametric floor slab with joists (*FS*) can be automatically generated between floors using the 2D polygon data representing a building footprint (Figure 4.32, Figure 4.33).

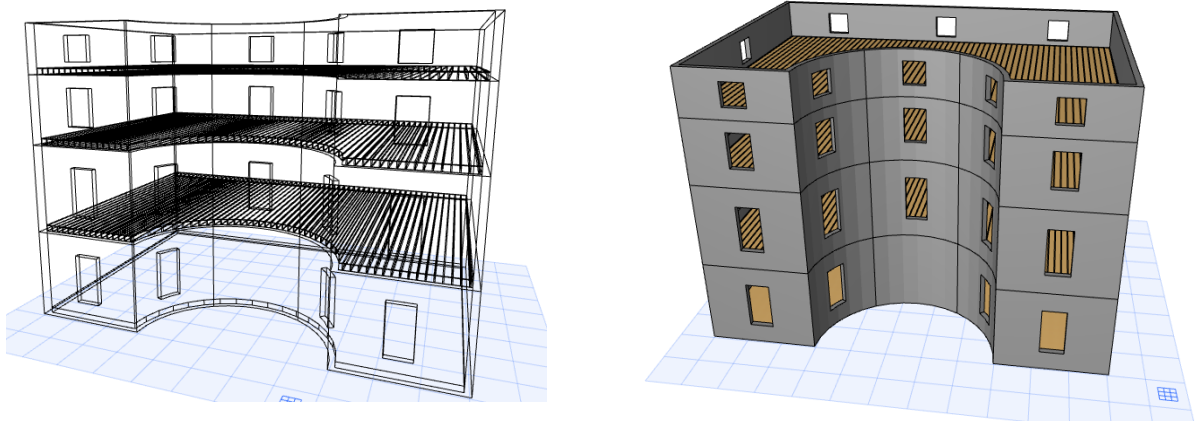


Figure 4.33: Replacement rule to automatically add objects such as a parametric floor slab between floors.

#### 4.4 Prototype II: Initial Implementation of Procedural Rules

The new procedural rules and algorithms in addition to the parametric vocabulary shapes for prototype II have been implemented as a plug-in for the ArchiCAD BIM software. Similar to the procedural façade prototype the Geometric Description Language (GDL) has again been used to code rules and algorithms described in the previous section. The C++ programming language has also been used to implement additional rules combined with an Application Programming Interface (API) from Graphisoft that can be used to extend the existing functionality of the ArchiCAD BIM software. The basic vocabulary shapes which incorporate the parametric HBIM library objects are all coded with GDL. The rules and algorithms for combining and procedurally generating geometry from the vocabulary shapes are coded with both GDL and C++ using the Graphisoft API. The GDL files and rules coded in C++ are packaged

as a Dynamic Link Library (DLL) that enables it to be added as a plug-in to the ArchiCAD BIM software.

The first extrude rule (Figure 4.2) described in the previous section was implemented using C++ with the Graphisoft API and GDL. In order to implement this rule, direct integration with existing ArchiCAD functionality was required which cannot be achieved using GDL on its own. The implementation of this procedural extrusion rule allows the coordinates from polygons representing building footprints to be retrieved and inputted into a GDL script that performs further calculations and creates the required geometry. Microsoft Visual C++ 2010 Express was used to write and compile code using C++. Table 4.9 shows the C++ functions which were implemented for this prototype to retrieve polygon data within the ArchiCAD environment. Functions one to three were written specifically for this prototype while functions four to seven are required functions for any plug-in to ArchiCAD. The steps for the main functions “*Do\_editLibraryObject*” and “*Do\_editLibraryObject\_UserInput*” are shown in Table 4.10 and Table 4.11.

Table 4.9: C++ Functions for implementing rule one – Procedural Extrusion

<b>Function 1:</b>	Do_editLibraryObject();
<b>Function 2:</b>	Do_editLibraryObject_UserInput();
<b>Function 3:</b>	GetSelectedElementPolygon();
<b>Function 4:</b>	CheckEnvironment();
<b>Function 5:</b>	RegisterInterface();
<b>Function 6:</b>	Initialize();
<b>Function 7:</b>	FreeData();

Table 4.10: Steps for function one “*Do\_editLibraryObject*” shown in Table 4.9.

1)	Retrieve and store selected polygon data by calling function three “ <i>GetSelectedElementPolygon</i> ”.
2)	Retrieve the global unique identifier (GUID) for the library part resource which is to be edited.
3)	Change the default parameters of the library part resource. This updates parameters of the GDL file with parameters of the selected polygon.
4)	Create an instance of the library object in the database and place on floorplan.

Table 4.11: Steps for function two “*Do\_editLibraryObject\_UserInput*” shown in Table 4.9.

1)	Create a polygon with user input and store results.
2)	Retrieve the global unique identifier (GUID) for the library part resource which is to be edited.
3)	Change the default parameters of the library part resource. This updates parameters of the GDL file with parameters of the created polygon.
4)	Create an instance of the library object in the database and place on floorplan.

Within ArchiCAD, these two functions are accessed from a new HBIM menu (Figure 4.34, Figure 4.35). The remaining steps in the procedural extrusion rule (Figure 4.4) are implemented with GDL after the polygon data is retrieved and passed to the GDL script. The retrieved polygon data is stored in arrays in the GDL script and used in subsequent steps. All remaining calculations for this rule are implemented in a master script which is the first GDL script to be executed. This includes converting arc angles from radians to degrees, calculating arc origins, arc start and end angles, arc radii, polygon area and polygon centroid coordinates. Finally, a mass model (*MM*) which is a vocabulary shape is generated using a *GDL* function which creates a 3D shape from a 2D polygon definition. This is coded in the 3D script with the vocabulary shape stored in a subroutine (Figure 4.37) that is called from an executive script at the start of the 3D script (Figure 4.36).

The offset algorithm for the second extrude rule (Figure 4.8) is implemented in the GDL master script. This involves calculating perpendicular directions for each building side, calculating offset lines and arcs and calculating intersections between offset lines and arcs. The results of the offset algorithm are the coordinates of the offset polygon which are stored in two arrays. These coordinates are then used to create an opening in the original mass model. The *cPRISM* function which is used to create a mass model supports a hole definition so the offset polygon coordinates can be directly inputted into the same *cPRISM* function to create the opening. A parameter for the level of detail (LoD) is used to apply this rule (Figure 4.36).

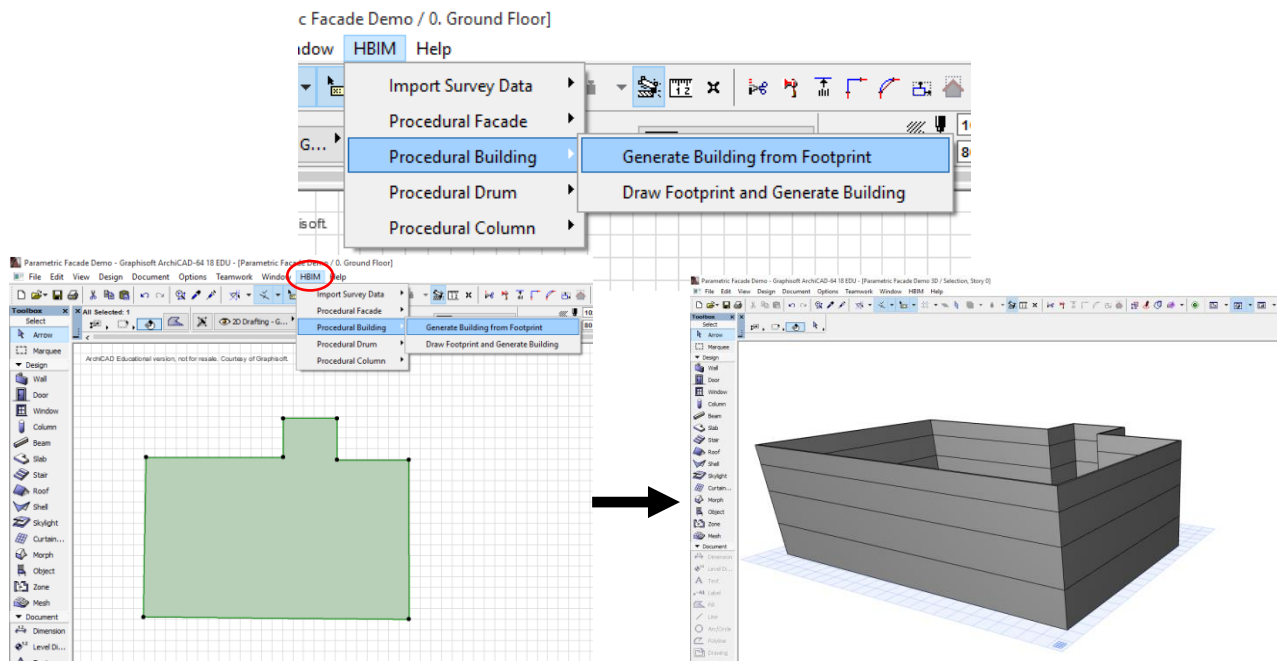


Figure 4.34: New HBIM menu to access functions of prototype plug-in (top). A selected 2D footprint (bottom left) is used to generate a building model (bottom right).

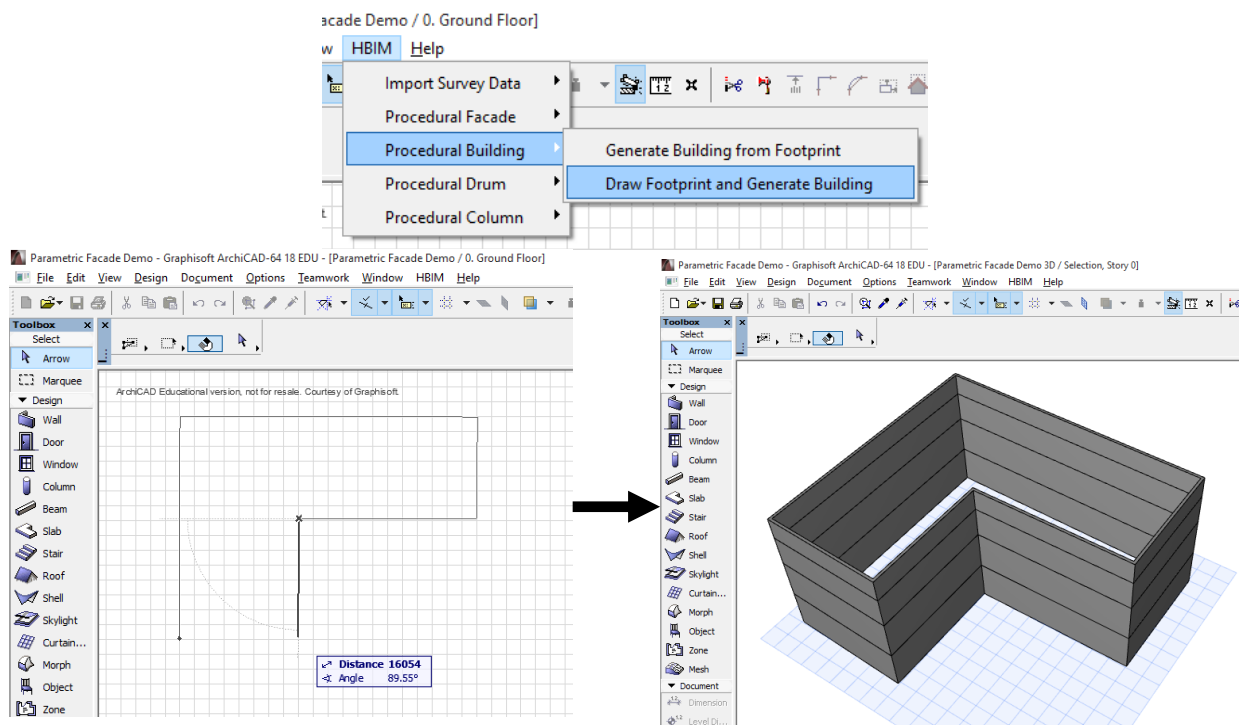


Figure 4.35: New HBIM menu to access functions of prototype plug-in (top). A building footprint is drawn by a user (bottom left) which automatically generates a building model (bottom right).

```

45  ! =====
46  ! Executive Script:
47  ! =====
48  ! .....
49  ! -----
50  !!! Building Walls & Hotspots:
51  ! -----
52  GROUP "WALL"
53  FOR storey = 1 TO nStoreys STEP 1
54  GOSUB 7000: !!! HOTSPOTS FOR fl_height
55  IF lod = 1 THEN
56  GOSUB 1000: !!! C_PRISM WITHOUT HOLE (BLOCK MODEL)
57  ADDz fl_height[storey]
58  ELSE
59  GOSUB 2000: !!! C_PRISM WITH HOLE (BUILDING WALLS)
60  ADDz fl_height[storey]
61  ENDIF
62  NEXT storey
63  DEL nStoreys
64
65  GOSUB 3000: !!! HOTSPOTS FOR BLOCK/BUILDING WALLS
66
67  ENDGROUP

```

Figure 4.36: Sample GDL code showing an executive script (located at the top of a 3D script) used to call subroutines for generating block models or wall models.

The repeat rule (Figure 4.14) is implemented in GDL using loop commands to repeat instances of the wall or mass model created with the previous rules. A user parameter for the “Number of Storeys” controls the number of iterations for this loop. The heights of each floor are first automatically calculated with classical proportions in the master script. These heights can subsequently be edited by a user in a dialogue box or graphically in 2D or 3D windows. The final stage of the repeat rule (Figure 4.18) is also executed in the GDL master script where the number of floors is calculated based on the updated building height during interactive editing (Figure 4.38).

```

392 ! =====
393 !!! SUBROUTINES:
394 ! =====
395
396 ! -----
397 1000:! C_PRISM WITHOUT HOLE (BLOCK MODEL)
398 ! -----
399 MATERIAL wmat          !!! Material
400 PUT bldg_x[1][storey], bldg_y[1][storey], 15 - arcStatusCode[1][storey]
401
402 IF arcAngle_wall[1][storey] <> 0 THEN
403   PUT arcOrigin_x[1][storey], arcOrigin_y[1][storey], 900 + 15,
404   | 0, arcAngle_wall[1][storey], 4000 + 15
405 ENDIF
406
407 IF PolygonArea >= 0 THEN      !!! IF polygon direction is counter-clockwise
408   FOR side = nBldg_sides TO 2 STEP -1
409     PUT bldg_x[side][storey], bldg_y[side][storey], 15 - arcStatusCode[side][storey]
410
411     IF arcAngle_wall[side][storey] <> 0 THEN
412       PUT arcOrigin_x[side][storey], arcOrigin_y[side][storey], 900 + 15,
413       | 0, arcAngle_wall[side][storey], 4000 + 15
414     ENDIF
415   NEXT side
416
417 ELSE                          !!!! IF polygon direction is clockwise
418   FOR t = 1 TO nBldg_sides - 1 STEP 1
419     PUT bldg_x[1+t][storey], bldg_y[1+t][storey], 15 - arcStatusCode[1+t][storey]
420
421     IF arcAngle_wall[1+t][storey] <> 0 THEN
422       PUT arcOrigin_x[1+t][storey], arcOrigin_y[1+t][storey], 900 + 15,
423       | 0, arcAngle_wall[1+t][storey], 4000 + 15
424     ENDIF
425
426   NEXT t
427 ENDIF
428
429 PUT bldg_x[1][storey], bldg_y[1][storey], -1
430
431 cPRISM_ "wmat", "wmat", "wmat", (NSP/3), fl_height[storey], get (nsp)
432
433
434 RETURN
435

```

Figure 4.37: Sample GDL code showing a subroutine which generates geometry for a mass model vocabulary shape (*MM*) using a *cPRSIM* function.



```

2353 ! =====
2354 ! 800: ALGORITHM 3: INTERACTIVE EDITING FOR NUMBER OF STOREYS
2355 ! =====
2356
2357 IF nStorey_edit = 0 THEN
2358   FOR storey = 1 TO nStoreys
2359     IF storey = 1 THEN
2360       bldg_height = fl_height[storey]
2361     ELSE
2362       bldg_height = fl_height[storey]+bldg_height
2363     ENDIF
2364   NEXT storey
2365
2366   PARAMETERS bldg_height = bldg_height
2367
2368   ENDIF
2369
2370   !-----
2371
2372   IF nStorey_edit = 1 THEN
2373
2374     !!-----
2375     !!!! Find floor that bldg_height is on and set nStoreys to this floor
2376     !!-----
2377
2378     find_floor = 0
2379     lowerFloorCheck = 1
2380     upperFloorCheck = 2
2381
2382     IF bldg_height > 0 AND bldg_height <= fl_height[1] THEN
2383       find_floor = 1
2384       PARAMETERS nStoreys = 1
2385       bldg_height = fl_height[1]
2386     ELSE
2387
2388       REPEAT
2389
2390         !-----
2391         !!! Break loop if nStoreys goes beyond fl_height array definition:
2392         !-----
2393         IF vardim1(fl_height) = upperFloorCheck THEN
2394           find_floor = 1
2395         ELSE
2396
2397           !-----
2398           !!! Calculate floor height below current iteration:
2399           !-----
2400           FOR storeyLoop = 1 TO (upperFloorCheck-1) STEP 1
2401             IF storeyLoop = 1 THEN
2402               sum_fl_height_below = fl_height[storeyLoop]
2403             ELSE
2404               sum_fl_height_below = fl_height[storeyLoop] + sum_fl_height_below
2405             ENDIF
2406           NEXT storeyLoop
2407
2408           IF bldg_height > fl_height[lowerFloorCheck] AND bldg_height <= (fl_height[upperFloorCheck]+ sum_fl_height_below) THEN
2409             find_floor = 1
2410             PARAMETERS nStoreys = upperFloorCheck
2411
2412             !-----
2413             !!! Set new bldg_height:
2414             !-----
2415             FOR storey = 1 TO nStoreys
2416               IF storey = 1 THEN
2417                 bldg_height = fl_height[storey]
2418               ELSE
2419                 bldg_height = fl_height[storey] + bldg_height
2420               ENDIF
2421             NEXT storey
2422
2423             ENDIF
2424             lowerFloorCheck = lowerFloorCheck + 1
2425             upperFloorCheck = upperFloorCheck + 1
2426
2427             ENDIF
2428             UNTIL find_floor = 1
2429
2430           ENDIF
2431         ENDIF
2432

```

Figure 4.38: GDL code to calculate the new number of floors when a building height has been graphically edited by a user.

The fourth procedural rule, a split rule is also implemented using GDL. One of the main challenges with implementing the split rule for this prototype is in the storage of parameters. Each object that is repeated or created during a split rule requires a new set of parameter values to enable individual object editing. In order for the procedural rules to be applied to many building arrangements, it requires parameters for tiles and tile objects to be extensible for any number of tiles on a floor, any number of floors and any number of building sides. This poses a number of challenges with regard to storing undefined numbers of variables. Tile and tile object parameters also need to be referenced to the building side, floor and tile that they are positioned on. Completely new tile and tile object parameters also cannot be created by a GDL script as hotspot editing requires parameters to be manually placed in the parameter dialogue.

A parameter with an undefined number of values, however, can be achieved with GDL using arrays. Using arrays it is possible to store multiple values for a single parameter in a table structure with specific parameter values referenced by the row and column position in that array. A one-dimensional array uses only one column with any number of rows while a two-dimensional array may use any number of rows and columns. Using two-dimensional arrays it is possible to allow parameter values to be extended based on two scenarios by storing new parameter values in new columns and rows.

In the first procedural façade prototype, this was used to store multiple parameter values for tile parameters. Table 4.12 shows an example of this where multiple values are stored in a two-dimensional array for a window width parameter. The row number relates to the tile that the window width represents while the column relates to the floor that the window width represents (Table 4.12). This allows parameters to be created and stored for any number of tiles and floors on a façade as any additional number of rows and columns can be added to the array. A specific window width is referenced in this

array by including the row number and column number after the parameter, e.g. Window\_Width(2)(3) references the window width in row two (tile two) and column three (floor three) which is 1.35 in Table 4.12.

Table 4.12: Example of a two-dimensional array for storing multiple values of a window width parameter.

<b>Window_Width</b> (Row = Tile Number, Column = Floor Number)				
	<b>Column 1</b> (Floor 1)	<b>Column 2</b> (Floor 2)	<b>Column 3</b> (Floor 3)	<b>Column 4</b> (Floor 4)
<b>Row 1</b> (Tile 1)	1.3	1.2	1.6	
<b>Row 2</b> (Tile 2)	1.29	1.3	<b>1.35</b>	
<b>Row 3</b> (Tile 3)	1.25	1.25	1.3	
<b>Row 4</b> (Tile 4)	1.31		1.3	

This method of storing an undefined number of parameter values works well for the first procedural façade prototype which contains a single façade. Prototype II, on the other hand, can have any number of building sides so parameters also need to be extensible for this third scenario. To overcome this, a new method of storing parameter values is adopted for the split rule with this prototype. Instead of just using column and row numbers to reference the location of an object, additional reference parameters values are stored with a tile or tile object parameter value. For example, when a window width parameter value is stored, two additional parameter values will also be stored in that array to reference the floor and building side that the window width relates to (Table 4.13). The tile position for each window parameter value is referenced by the row in the array (Table 4.13). With this method, three columns are used for each floor on a particular building side. This new storage method enables the number of tiles on a floor to be extensible using new rows and the floor and the building sides are both extensible using new columns.

When a split rule is applied to a particular floor, the relevant array table for storing the new parameters is first checked to see if any three columns have already been assigned to that floor. If they have, then the new set of parameters will replace the previous

parameters as a new number of tiles are being generated for that floor. If three columns in an array have not been previously assigned to that floor then the next three available empty columns are used. For each tile that is being created on a particular floor, three parameter values are stored, the building side, the floor number and the window width values for that tile (Table 4.13) (Figure 4.39).

Table 4.13: Example of new method for storing extensible parameter values in arrays for implementing procedural rule four-Split Rule.

Window_Width									
Tile (Row)	Side	Floor	Window width	Side	Floor	Window width	Side	Floor	Window width
	1	2	3	4	5	6	7	8	9
1	1	1	1.3	1	2	1.28	3	1	1.35
2	1	1	1.25	1	2	1.22	3	1	1.22
3	1	1	1.35	1	2	1.33	3	1	1.31
4	1	1	1.28	1	2	1.24	3	1	1.29
5							3	1	1.5
6									

When accessing parameters from this table it is possible to find any required parameter value by searching columns based on the reference parameters for the building side and floor. However, to avoid unnecessary searches, every time a floor is split and three new columns are assigned to a floor, a reference to the column position in the array is stored as a completely separate parameter. This is like a pointer which points to the location of a value in an array. For example in Table 4.13, when the second floor of building side one is split, the parameters are stored in columns four, five and six. After these parameter values are created a completely separate array parameter also stores the location of the first column for this floor and building side. For this example the new parameter for the reference location would be  $\text{Window\_Width\_Column}(1)(2) = 4$  where the row one (1) references building side one and column two (2) references floor two and the value four represents the location column in the Window\_Width array. This

greatly speeds up the process of accessing parameters for this procedural rule and avoids unnecessary searches.

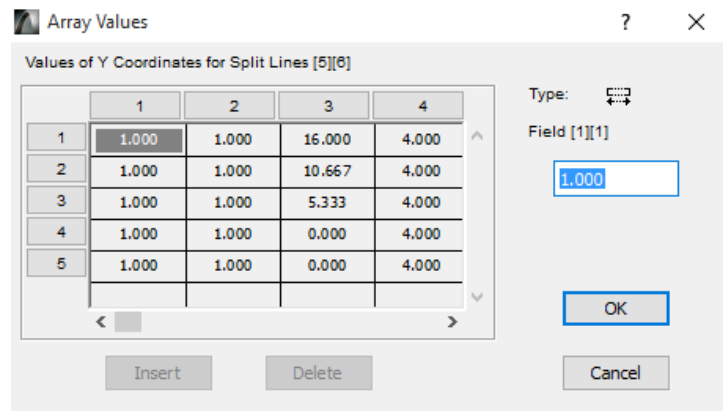


Figure 4.39: Example of an extensible two-dimensional array in ArchiCAD for storing splitLine\_y coordinates.

All arrays for the storage of parameters for this rule are setup in a master script with GDL. Calculations for this procedural rule (Figure 4.21) are also implemented in the master script. This includes calculating coordinates of split lines which divide a floor into tiles, calculating tile lengths and calculating window opening coordinates on tiles. Finally, the geometry required for this rule is implemented in the 3D script. Tiles are shown graphically by placing 3D lines using split line coordinates with a GDL *LIN\_* function. Window openings are cut from wall instances using Boolean operations. A GDL *cPRISM* function is used to define the position of openings which are then subtracted from wall instances. 3D lines showing tile splits and geometry used to cut window openings are stored in subroutines in the 3D script. An executive script at the top of the 3D script is used to call subroutines and includes relevant transformations for placing geometry.

The fifth rule, a replacement rule is implemented using the GDL master script and 3D script. Vocabulary shapes such as floor slabs or windows are stored in individual subroutines in the 3D script and called from an executive script at the top of the 3D

script. The master script is used to match parameters of a vocabulary shape to the tile or floor that they will be placed on. For example, the shape and size of a floor slab are automatically determined from the 2D polygon data and parameters for a window size are automatically populated based on window openings in a wall.

When a building model is generated with this prototype from a 2D building footprint all the rules are automatically executed in a sequence and do not require user interaction at each stage. Once a model is automatically generated users can then alter parameters to change the building structure or library objects as required. An example of different building models automatically generated with the procedural building prototype can be seen in Figure 4.40. Different numbers of floors and openings on a floor are generated by altering parameters.

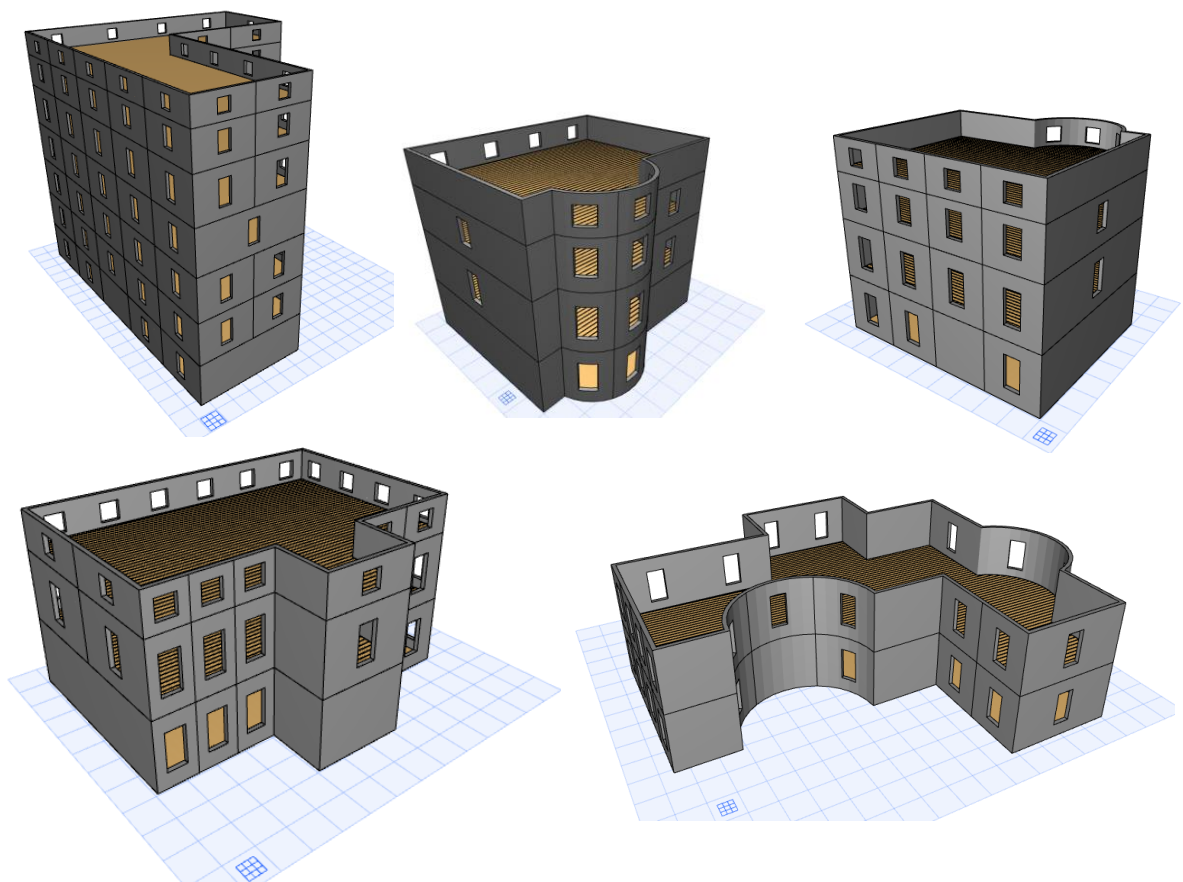


Figure 4.40: Various building models automatically generated with the rules and algorithms for the procedural building prototype II.

Procedural rules can also be applied to multiple building footprints at once. Figure 4.41 to Figure 4.43 show an example of this where building footprints for an area of Dublin city centre are used to automatically generate building models. An Ordnance Survey Ireland (OSI) dataset comprising of 3,610 building footprints were imported into the ArchiCAD software and the procedural modelling rules were applied to automatically generate building geometry from these footprints. The building footprints covered an area roughly 1.7 km in length and 1.4 km in width. Using 3D GIS software it is possible to extrude building footprints to create mass models but it is not possible to automatically generate more detailed parametric models containing walls, opening, floors etc. With the procedural building prototype, any parametric building can be altered to generate different numbers of floors and any tile arrangements with openings and objects. If required this generated geometry can be precisely mapped to survey data such as orthographic images of point clouds. Figure 4.43 shows building geometry exported in the standard Industry Foundation Class (IFC) format.



Figure 4.41: Parametric building models automatically generated from 2D building footprints for an area of Dublin city centre.



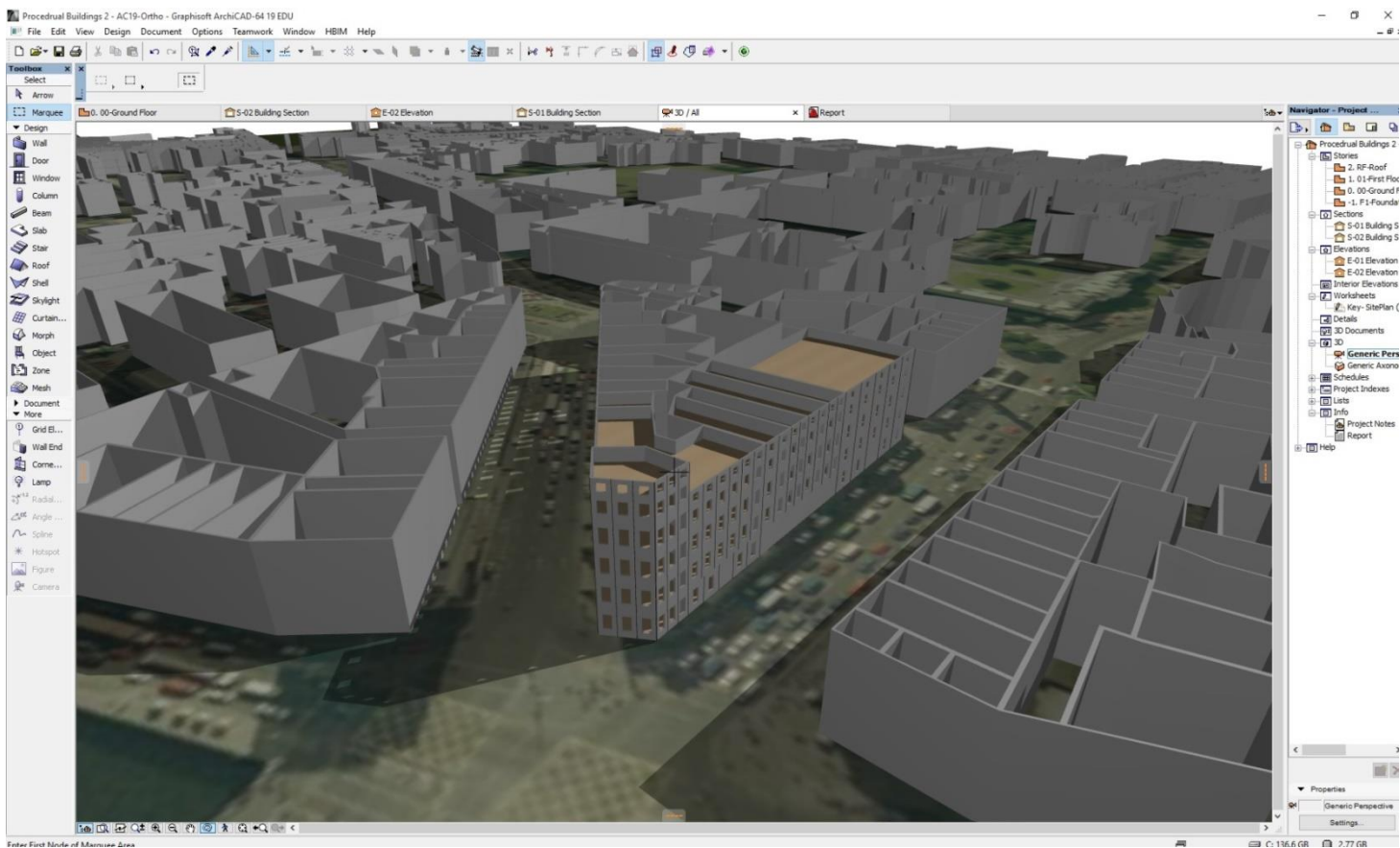
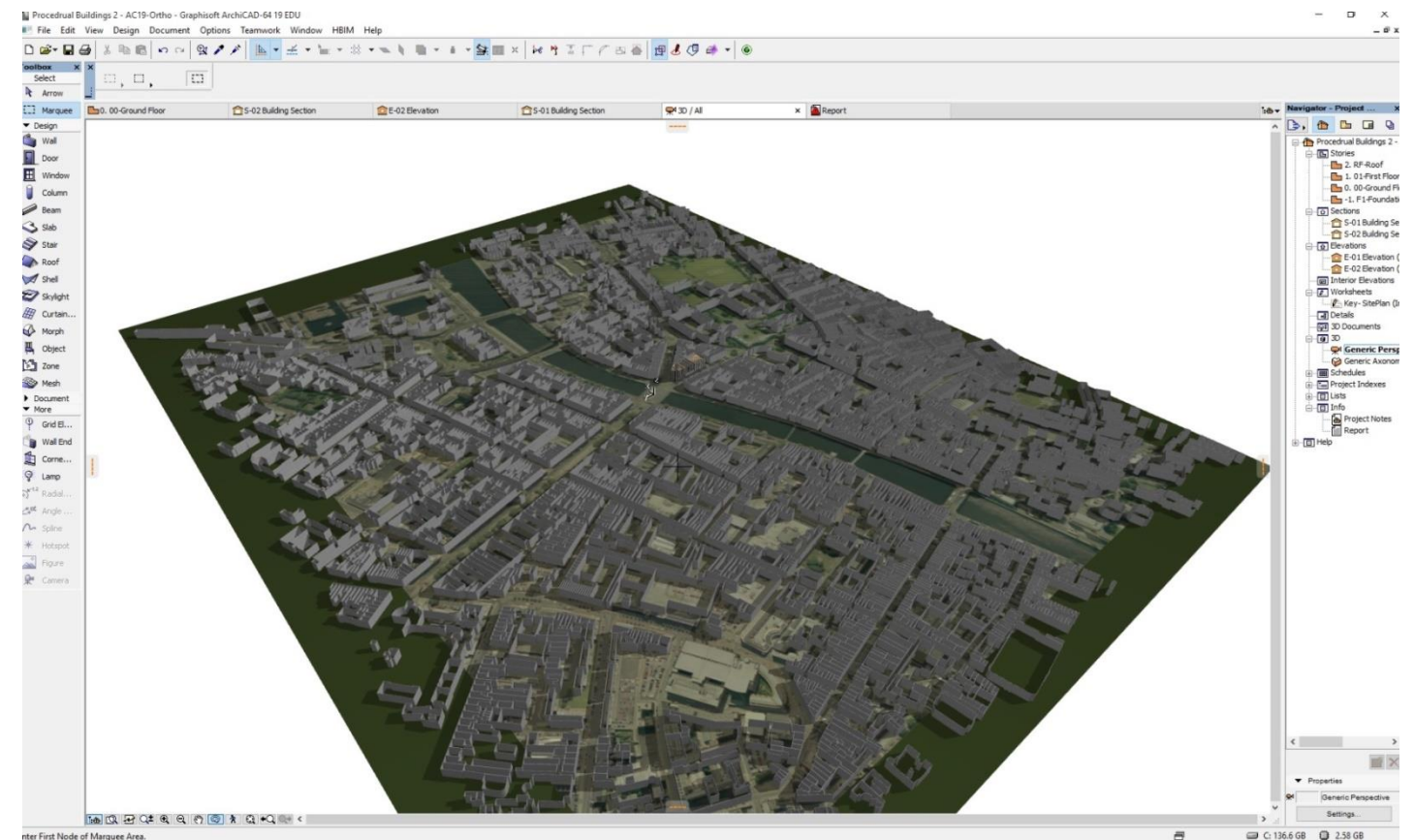


Figure 4.42: Parametric building models automatically generated from 2D building footprints for an area of Dublin city centre.



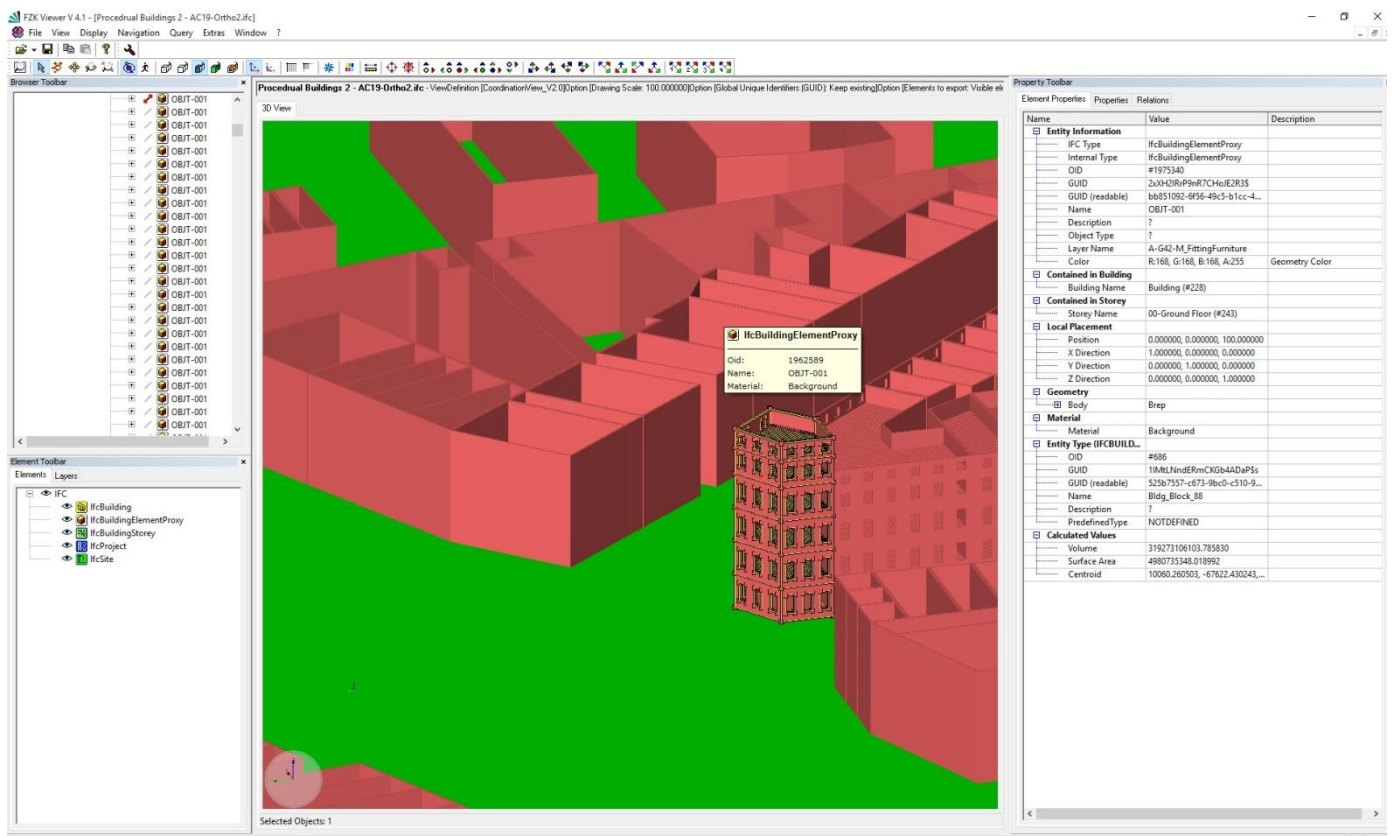
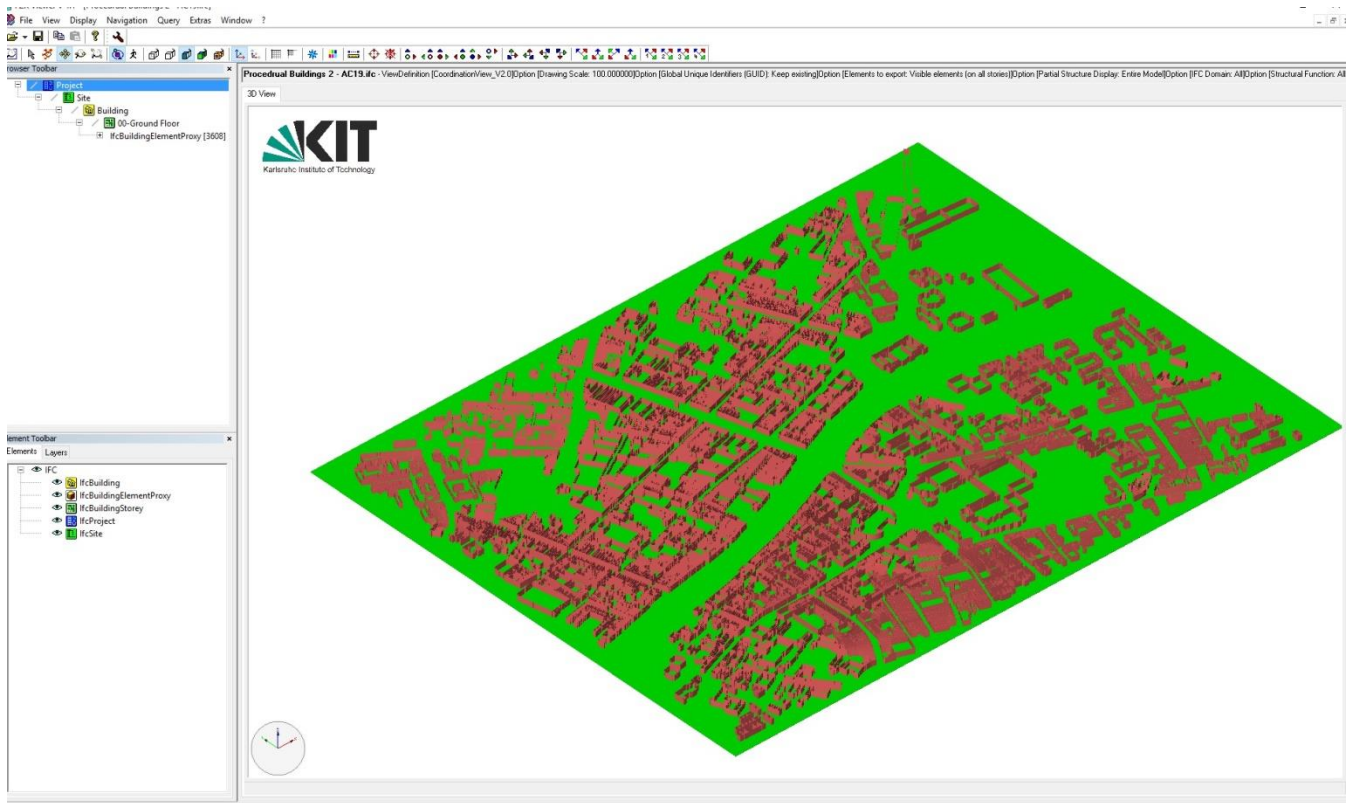


Figure 4.43: IFC building models automatically generated from 2D building footprints for an area of Dublin city centre.

A video showing a demonstration of the new procedural building prototype can be seen from the link below.

**<https://youtu.be/WJzxR5qm8xc>**

#### **4.5 Prototype II: Summary**

The procedural building prototype II further developed concepts from the first procedural façade prototype by providing capabilities for procedurally modelling all faces of a building and not just a single façade. With the procedural building prototype, any building shape can be generated from 2D building footprints. A new set of rules and algorithms have been designed and implemented to:

1. Generate mass models from 2D building footprints.
2. Convert mass models to higher level of detail models containing walls.
3. Repeat walls for any number of storeys.
4. Split a floor, façade or building into tiles containing openings.
5. Replace and add new vocabulary shapes to previously generated geometry.

Classical architectural rules and proportions also assist with the generation process by providing an initial estimate for the size and positioning of objects on a façade. These new rules and algorithms have been implemented as a plug-in to the ArchiCAD BIM software using the Geometric Descriptive Language (GDL) and the C++ programming language with an Application Programming Interface (API) for the ArchiCAD software. These new rules and algorithms for the procedural building prototype enable the modelling of existing buildings with a semi-automatic process where the required geometry is first generated and then manually refined to match to specific survey data. The mapping of generated geometry to survey data is described in Chapter 6.

## 4.6 Prototype II: Limitations

The developed procedural building prototype can greatly improve the efficiency of modelling existing buildings by automatically generating building geometry and automatically combining and positioning objects on this building geometry. A limitation of this prototype, however, is that it has limited capabilities for modelling irregular geometries caused by deformation. Historical buildings often contain irregular geometries caused by damage and environmental conditions over time. This may include non-orthogonal walls that do not intersect at 90-degree angles or walls that are not perfectly vertical. The procedural building prototype provides limited capabilities for representing the true condition of a building by enabling the modelling of non-orthogonal walls that do not intersect at perfect 90-degree angles. Non-uniform objects and openings can also be generated, for example, exact corners of window openings can be specified which do not have to be perfect 90-degree rectangles.

This prototype, however, cannot create walls that are off plumb or not perfectly vertical. With the first procedural façade prototype, an inclination angle could be set to model non-vertical walls. This is relatively straightforward for a single planar surface. The generation of non-vertical connected planar and curved building faces is considerably more complex. As the recording of historical buildings often requires such deformations to be documented, a new prototype has been developed to provide tools for more accurate generation of irregular building geometry containing deformation. This new prototype is described in Chapter 5.

## Chapter Five: Prototype III – Irregular Procedural Building

### 5.1 Introduction

In order to overcome limitations of the previous procedural building prototype, a third prototype has been developed to procedurally generate irregular building geometry which contains deformation. Current BIM software has very limited tools for accurately modelling irregular building geometry that is often found in existing and historical buildings. For this reason, there is a great need for new software tools that are capable of achieving these results. This third prototype provides capabilities for modelling non-vertical walls using multiple cut-sections from survey data. This chapter describes the conceptual design framework and initial implementation for Prototype III (Figure 5.1).

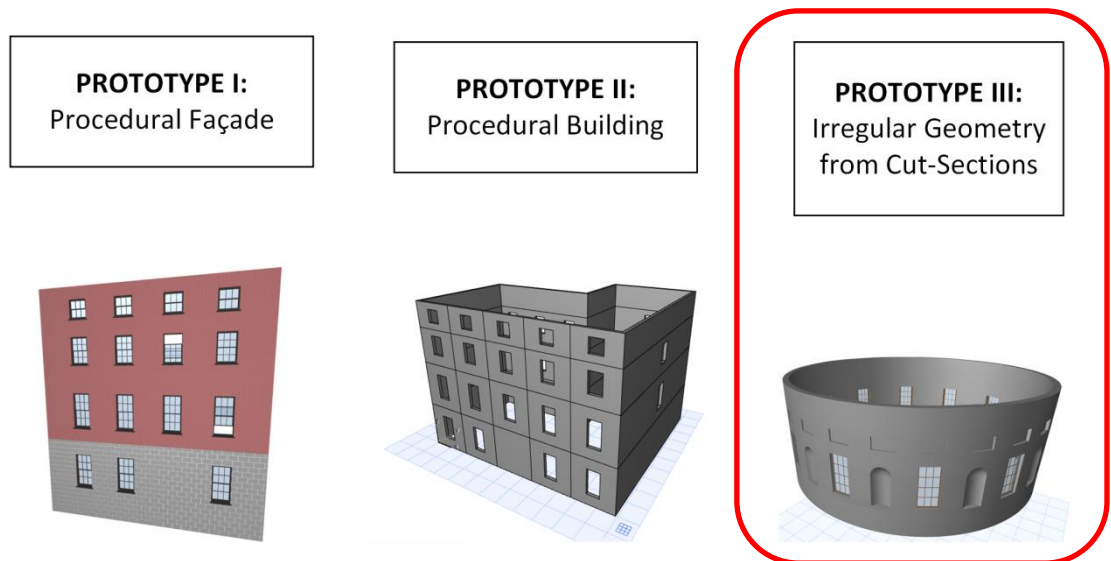


Figure 5.1: Prototype III for generating irregular building geometry containing deformation.

### 5.2 Overview of Prototype III: Irregular Procedural Building

Existing tools within BIM authoring software enable the modelling of non-vertical planar walls using a parameter for an inclination angle but do not provide tools for modelling non-vertical circular walls. The third procedural modelling prototype focuses

on procedurally generating non-vertical circular walls. Similar to the previous two prototypes, the third prototype also enables modelling buildings with a semi-automatic process where building geometry is first generated and then refined to match to survey data. This prototype has again been implemented for the ArchiCAD BIM software using the Geometric Description Language (GDL) and C++ with an Application Programming Interface (API) for ArchiCAD software.

### **5.3 Prototype III: Rule and Algorithmic Design**

The design and conceptual framework for the Irregular Procedural Building prototype is also based on concepts from shape grammars where a vocabulary of shapes is used with a set of rules and algorithms to automatically generate different building arrangements. The design for the Irregular Procedural Building prototype incorporates similar vocabulary shapes as the previous prototypes (Figure 3.8) but with additional new objects such as arch-top niches, rectangular niches and columns (Figure 5.25). A new set of procedural rules and algorithms have been designed which are described in this section. Section 5.4 then describes the initial implementation of these new procedural rules.

#### **5.3.1 Rule One: Procedural Surface Generation from Cut-Sections**

Rule one is a procedural surface generation from cut-sections (Figure 5.2). This rule is used for accurate and efficient modelling of curved wall geometry that contains deformation. This rule could be applied to a complete circular wall such as a cylindrical drum supporting a dome or to a particular side of a building footprint that contains curved wall geometry. As a result of deformation, such walls may not be perfectly vertical and may contain warping or deviation at different heights and locations. Unlike the previous prototype where a single footprint was used to generate a wall object, the

conceptual design for this prototype involves using multiple cut-sections at different heights to generate wall objects representing their true condition (Figure 5.3).

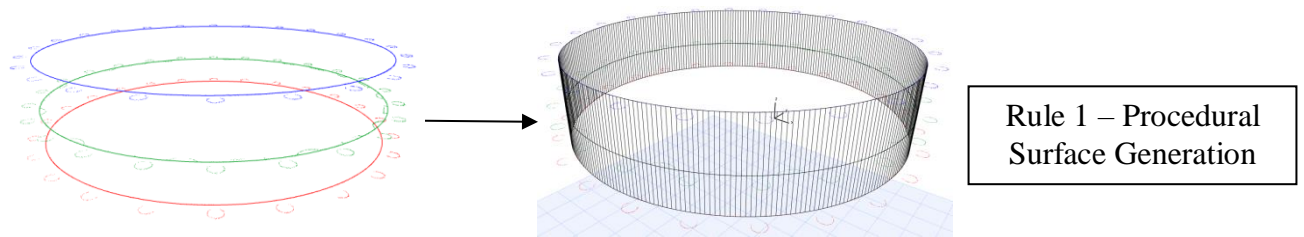


Figure 5.2: Rule One – Procedural surface generation from any number of horizontal cut-sections.

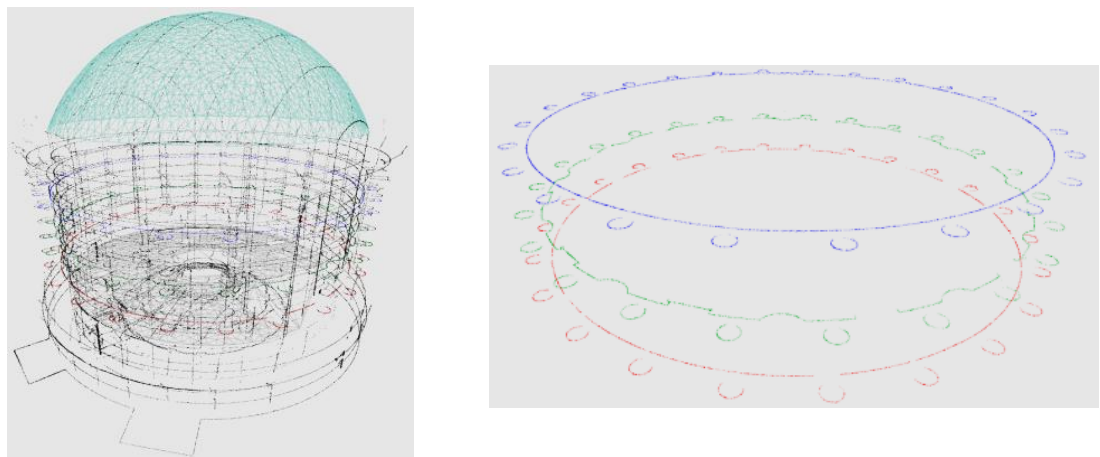


Figure 5.3: Multiple cut-sections through a point cloud used to model irregular and non-vertical wall geometry.

Creating a surface from cut-sections through a wall at different heights allows the variances of the surface to be accurately represented. This new procedural rule allows users to select any number of horizontal cut-sections which are then used to automatically generate an irregular wall surface. The non-vertical wall is created by generating a surface between each horizontal cut-section. This rule results in automatically generated geometry for a circular wall which represents the true condition of the wall with deformation based on scan data.

ArchiCAD and also ArchiCAD's scripting language GDL, have no direct functions that are capable of generating a non-vertical curved surface from multiple cut-sections. A GDL *RULED* function, however, does enable a shape to be created from two polygon definitions at different heights (Figure 5.4). The two polygons can have different outlines but must have the same number of nodes. Connected nodes are defined by the order of nodes. This enables non-vertical planar faces to be generated but it does not allow for arcs in a polygon definition so it cannot directly generate non-vertical curved surfaces (Figure 5.4).

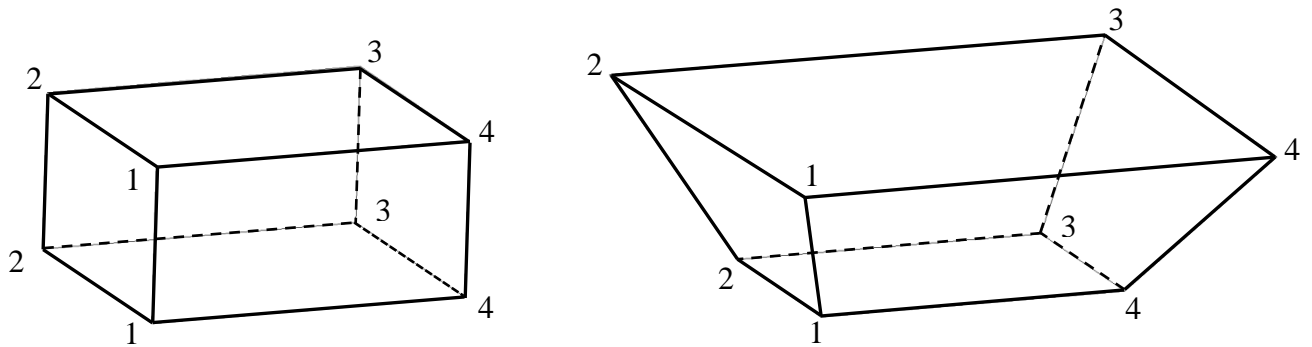


Figure 5.4: GDL *RULED* function which creates a shape that connects two polygons at different heights with the same number of nodes.

The design for this procedural rule is based on the GDL *RULED* function (Figure 5.4). Circular geometry is used with this function by estimating an arc in a 2D polygon by dividing it into a series of straight line segments. The number of straight line segments used to estimate a curve is defined by a parameter for the resolution. This enables non-vertical circular geometry to be created from polygon sections at different heights. Figure 5.5 shows an example of a GDL *RULED* function used to create non-vertical circular geometry by connecting two polygon definitions with the same number of nodes. Circular geometry is created with this *RULED* function by estimating a 2D arc or circle with a series of straight line segments.

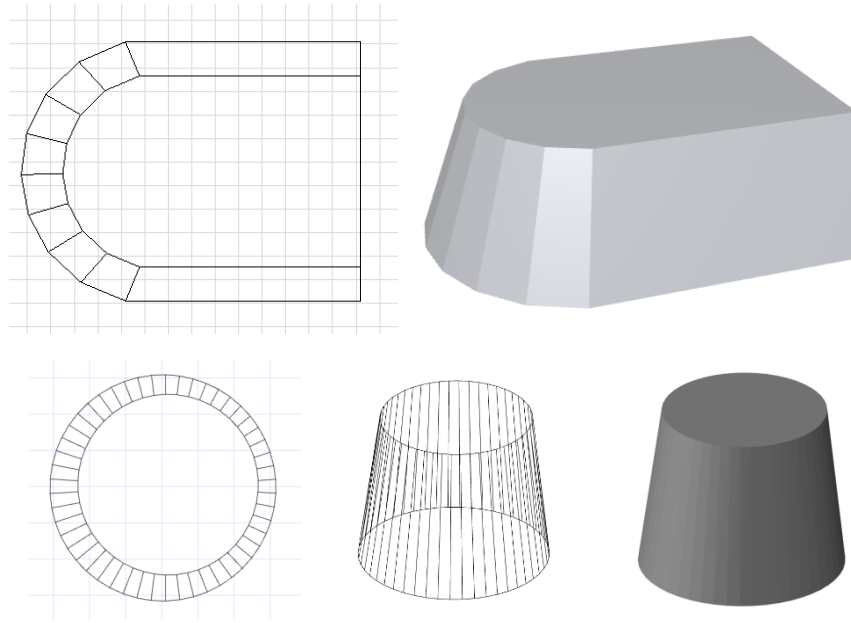


Figure 5.5: GDL *RULED* function used to create non-vertical circular geometry between two polygon definitions. Circular geometry created by estimating a 2D arc or circle with straight line segments.

The input for this rule is a set of horizontal cut-sections which are created from the point cloud. For this rule, the cut-sections need to be represented as closed polygons made up of any number of arcs. The conversion of a point cloud section into a closed polygon containing arcs can be automatically calculated in separate software. For example, the AutoCAD Civil 3D software from Autodesk can automatically calculate best-fit arcs from a series of points. Figure 5.6 shows a point cloud section through a circular wall (blue) and best-fit arcs (red) that were automatically fitted to the points using AutoCAD Civil 3D. In classical architecture, a circular wall such as a drum supporting a dome will be designed with a uniform diameter. In a historical building, however, the circle may no longer be a perfect circle as designed due to deformation or warping. For this reason, to represent the true condition of a wall each section is usually made up many separate connected arcs as opposed to one circle. In Figure 5.6 sixteen arc segments were fitted to the point cloud to accurately represent the shape of this section. Each of these arcs may have different centre points, radii and arc lengths.



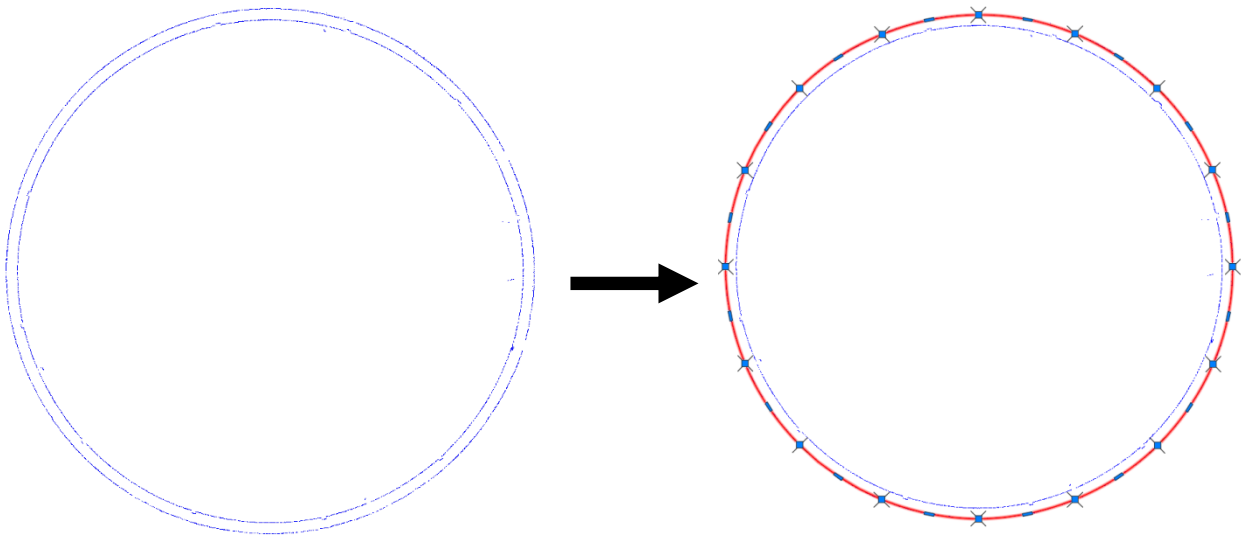


Figure 5.6: Point cloud section through a circular wall (left) and same point cloud section converted to a polygon containing sixteen arcs (right)

Figure 5.7 shows the inputs and outputs for this procedural rule. With this rule, users can select any number of polygon cut-sections to automatically generate a surface. The input data automatically extracted from selected polygon sections include the number of sections, height of sections, number of arcs per section, arc start and end coordinates and arc angles (Figure 5.7).

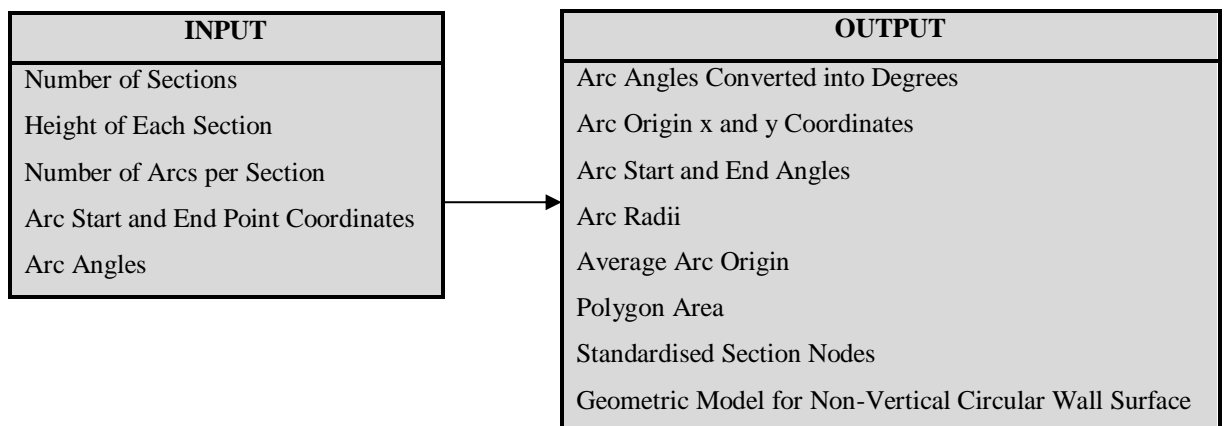


Figure 5.7: Inputs and outputs for rule one – Procedural Surface Generation

Figure 5.8 shows a workflow diagram for the steps involved in this procedural rule. A number of arc operations are initially performed to extract further data from arcs in sections. This includes converting arc angles from radians to degrees (Equation 4.1), calculating arc origins (Table 4.1), calculating arc start and end angles (Equation 4.3) and calculating arc radii (Equation 4.2). An average arc origin is also calculated between all arcs on all arc sections which is required for later calculations (Equation 5.1). The area of each polygon is also calculated to determine the polygon direction for each section (Equation 4.8).

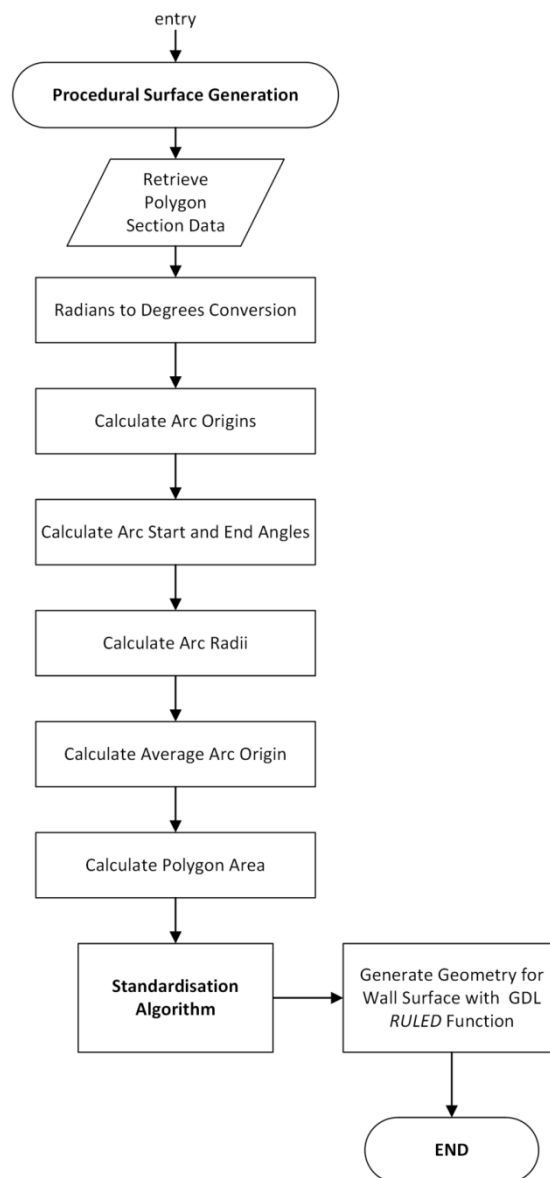


Figure 5.8: Flow diagram showing steps for rule one – Procedural Surface Generation

$$\begin{aligned}
avgArcOrigin_{x[section]} &= \left( \sum_{arcNumber=1}^{nArcs[section]} arcOrigin_x[arcNumber][section] \right) / nArcs[section] \\
avgArcOrigin_{y[section]} &= \left( \sum_{arcNumber=1}^{nArcs[section]} arcOrigin_y[arcNumber][section] \right) / nArcs[section] \\
avgArcOriginAllSections_x &= \left( \sum_{section=1}^{nSections} avgArcOrigin_x[section] \right) / nSections \\
avgArcOriginAllSections_y &= \left( \sum_{section=1}^{nSections} avgArcOrigin_y[section] \right) / nSections
\end{aligned}$$

Equation 5.1: Equations for calculating the average arc origin x and y coordinates for all arcs on all sections.

The main challenge involved with this procedural rule is in determining how points in one cut-section correspond and relate to points in the next cut-section above and below it. As all cut-sections will vary there are no common points between each section. There can also be a different number of arcs in each horizontal section. A *GDL RULED* command requires the same number of nodes in each section along with the nodes ordered to indicate connecting points. To overcome this problem a standardisation algorithm has been developed to regularise different sections with the same number of nodes and also to determine how points should connect between sections.

Figure 5.9 shows a diagram for the steps involved in this standardisation algorithm. The first part of this algorithm involves creating a framework of standard lines between each section. This framework is used to identify relating points between sections and also to calculate an equal number of coordinates along each section at regular intervals. This framework is created by defining radial lines from the average origin between all arcs in all sections (Equation 5.1) (Figure 5.10). The number of radial lines is dictated by a user parameter for the resolution. The resolution parameter can be any integer greater than four. A circular wall is split into a number of radial lines equal to the resolution integer. For example, if the resolution is set to 100 then 100 lines will be created for the

framework. A higher number of radial lines will result in a more accurate estimation of a curve and also a more detailed geometric model. This is because the standard radial lines are used to estimate a curved surface by calculating nodes at the intersection of these lines and the curved surface. The angles between these radial lines are calculated by dividing 360 (for a circular wall) by the resolution (Equation 5.2). Figure 5.11 shows another example of a standard framework created for a section comprising of four arcs. For illustrative purposes, four noticeable different arcs are shown in this figure to better differentiate between each arc.

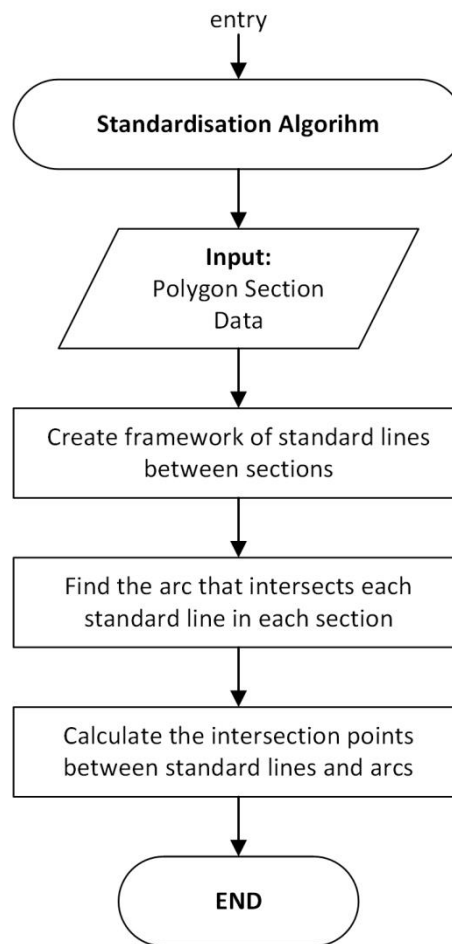


Figure 5.9: Flow diagram showing steps for the standardisation algorithm.

$$nNodes[section] = circleResol$$

$$standardAngle = 360/circleResol$$

Equation 5.2: Equation for calculating the angles between radial framework lines.

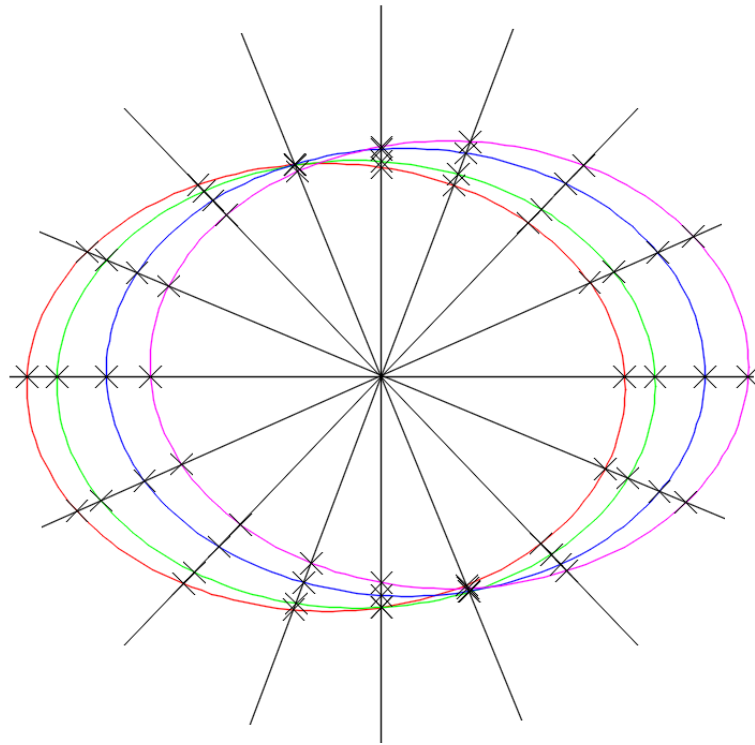


Figure 5.10: Diagram showing a framework of standard radial lines between four horizontal cut-sections.

After the framework of standard lines is set up, the next part of the workflow for this algorithm involves finding the arcs that intersect each standard line for each section. This is required for the final part of the workflow to calculate the intersection points between these arcs and standard lines. The arc that intersects a standard line is found by comparing angles of a standard line with the angles from the average arc origin to each arc start and end point (Figure 5.12). If the angle of a standard line falls within the angles from the average arc origin to an arcs start and end points then that arc intersect the line. Clauses also need to be added if an arc crosses 360 or 0 degrees. In this case, depending on the polygon direction, 360 degrees are added to angles from the average arc origin to either the start or end points. Once the number of the arc that intersects each standard line has been found, the final step involves calculating the intersection points between these arcs and standard lines. The intersection points between standard lines and arcs in each section are calculated using the steps in Table 5.1 as shown graphically in Figure 5.13.

Table 5.1: Steps for calculating the intersection points between standard radial lines and arcs in each section (Figure 5.13).

1)	Calculate internal angle marked “A” of the triangle in Figure 5.13 by subtracting the angle of the standard line from the angle of the line between origins.
2)	Calculate the second angle marked “B” of the triangle in Figure 5.13 using the sine rule (Equation 5.3).
3)	Calculate the final angle marked “C” of the triangle in Figure 5.13 by subtracting previous angles from 180.
4)	Calculate the distance to the intersection point from the average arc origin using the sine rule (Equation 5.4).
5)	Calculate the coordinates of the intersection point using the calculated distance and angle from the average arc origin (Equation 4.7).

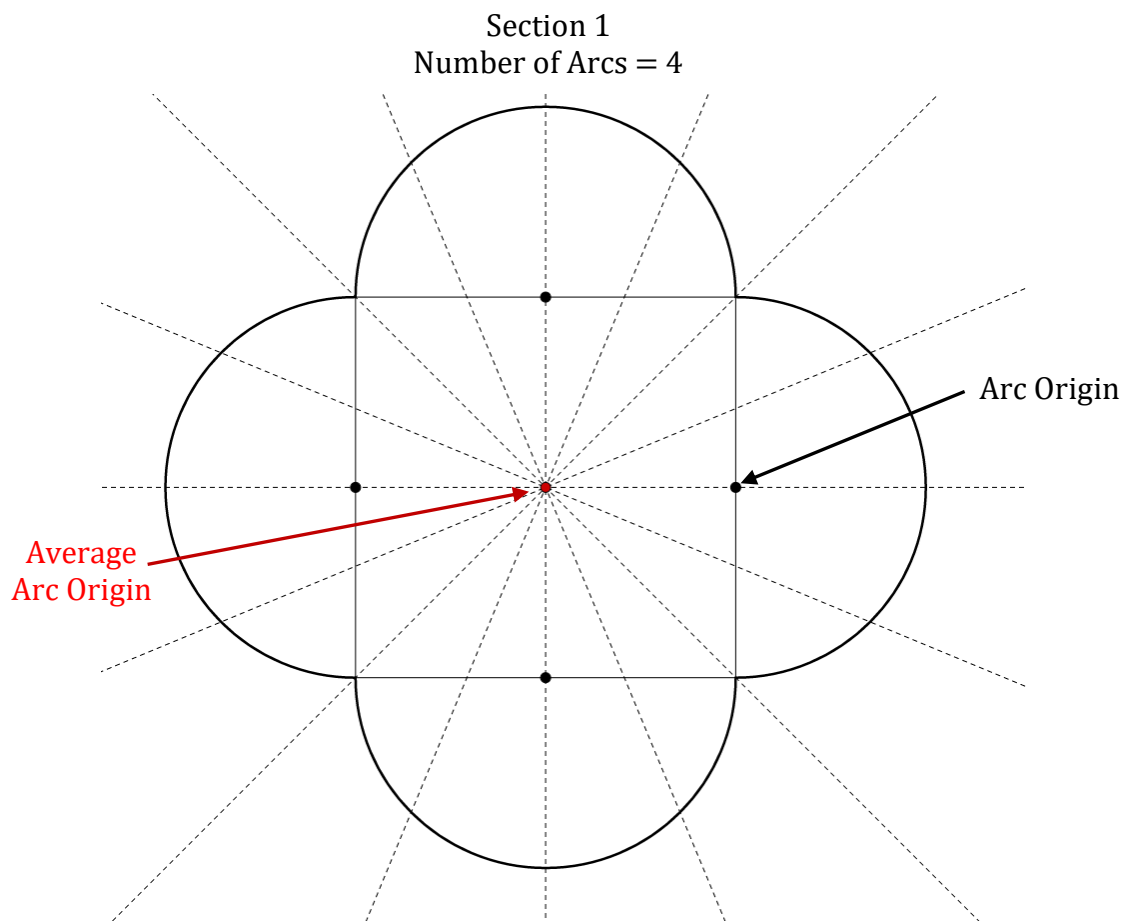


Figure 5.11: Diagram showing framework of standard radial lines created for a section comprising of four arcs.

With this solution, a clause is required if the angle of the standard radial line has the same angle as the line between an arc origin and average arc origin. In this case, the distance to an intersection point is calculated by adding the distance between the average arc origin and an arc origin to the radius distance of that arc. This distance

along with the angle of the standard line is then used to calculate the intersection coordinates using Equation 4.7.

The intersection points are calculated for each standard line on each section. Sections at different heights are connected by joining intersection points between different sections on the same standard line (Figure 5.10). This is because a line from the average arc origin to a point on a section is an estimate for the average normal direction of curvature between sections.

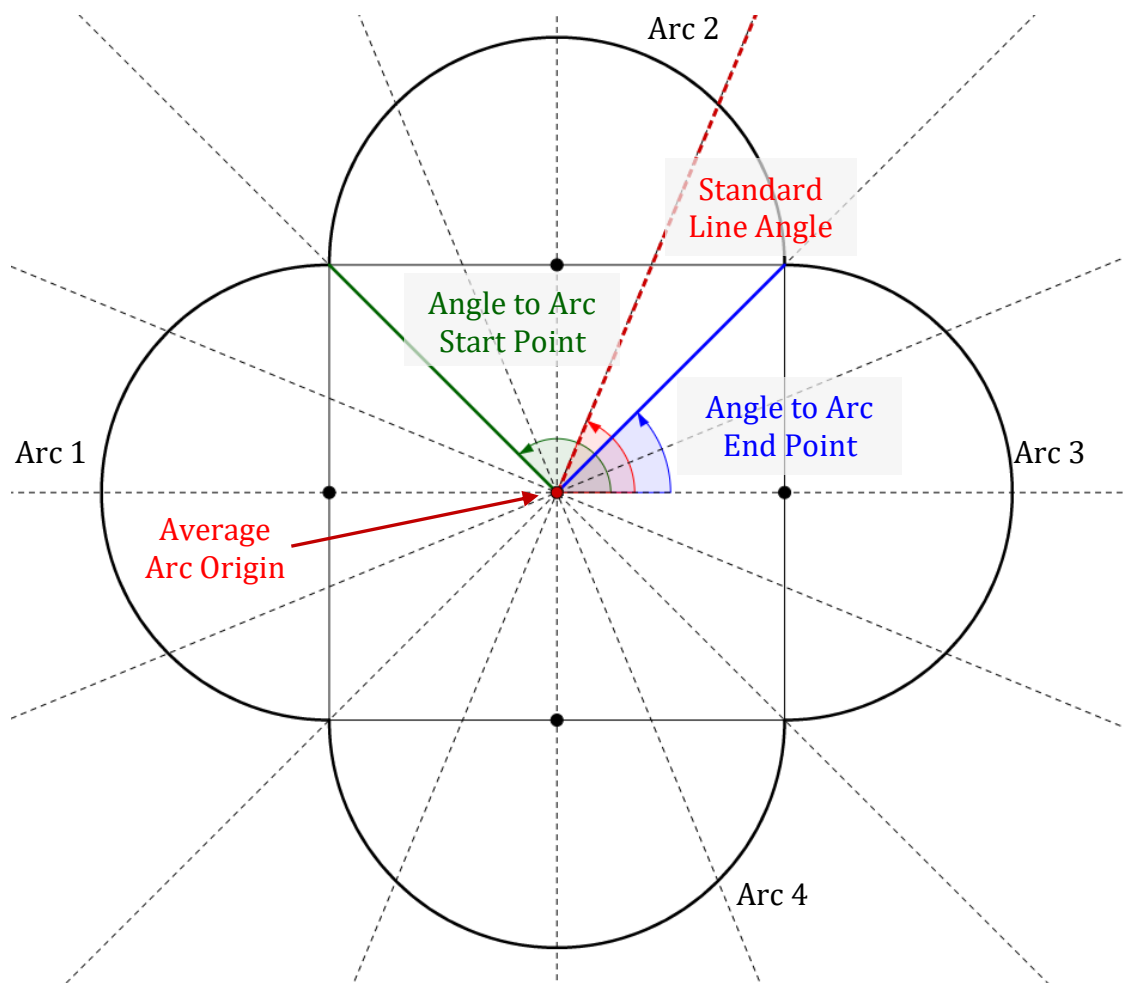


Figure 5.12: Diagram showing the solution for finding which arcs intersect with each standard radial line.

The final step in the workflow for this procedural rule involves generating the geometry for a non-vertical circular wall surface. Instead of using retrieved arc data to create geometry, the standardised nodes are instead used which estimate the curves originally defined using arcs (Figure 5.14). These standardised nodes enable the GDL *RULED*

command to be used to create a surface which connects each section. The GDL *RULED* command can be used as all sections now have the same number of nodes and are ordered by the nodes which connect different sections (Figure 5.10).

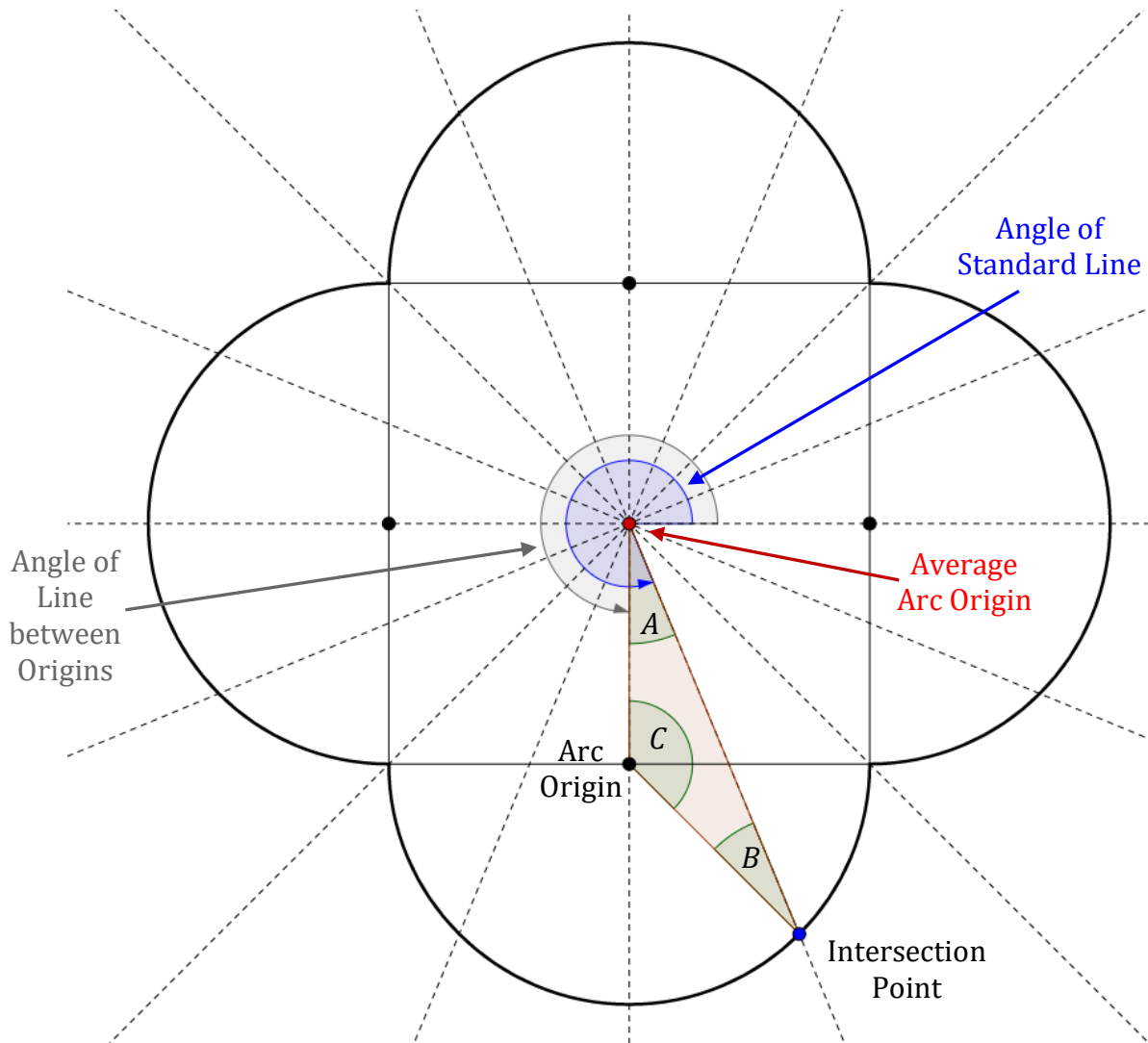


Figure 5.13: Diagram showing solution for calculating the intersection points between standard radial lines and arcs in each section.

$$B = \sin^{-1} \left( \frac{(\sin A \times (\text{distance between origins}))}{\text{arc radius}} \right)$$

Equation 5.3: Sine rule used to calculate angle B in Figure 5.13.



$$distance\ to\ intersection\ point = \left( \frac{(\sin C \times arc\ radius)}{\sin A} \right)$$

Equation 5.4: Sine rule used to calculate the distance from the average arc origin to the intersection point in Figure 5.13.

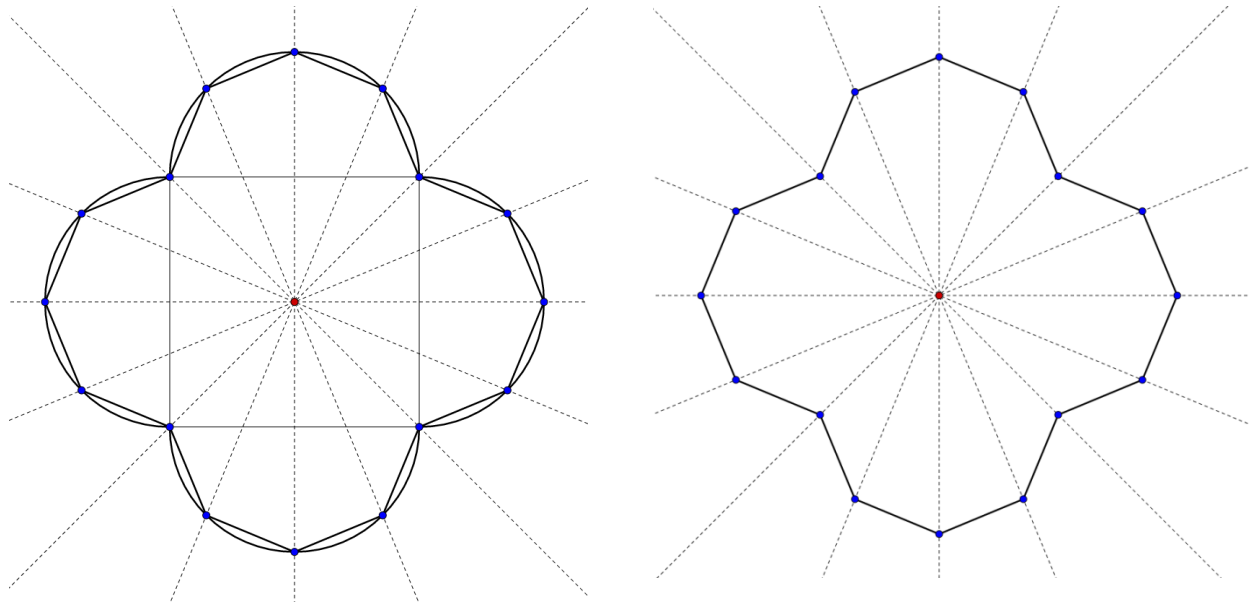


Figure 5.14: Diagrams showing a low-resolution estimate of 2D arcs by straight line segments which connect standardised nodes.

### 5.3.2 Rule Two: Offset Rule

The second procedural rule for this prototype is an offset rule. This is used to generate a higher level of detail model that contains both internal and external wall surfaces. Users can create an internal wall surface in two ways. For high accuracy results, users can reapply the first procedural rule to generate the internal wall surface from cut-sections. Boolean operations are then used to subtract the new internal wall surface from the external wall surface leaving just the area contained by the wall. This results in the true condition of both internal and external wall surfaces showing any deformation or deviations as surveyed from scan data.

Alternatively, if cut-sections are not available for the internal wall surface or if the wall thickness is believed to be uniform, a user can apply the second procedural rule which offsets the irregular external wall surface by a uniform thickness. The offset distance or wall thickness is a parameter for this rule. Unlike the previous prototype, the offsetting of a wall surface with this prototype is less complex and does not require a lot of calculations. This is due to the fact that the retrieved arc data for each section has previously been regularised with a regular interval of nodes between all sections that are created from a common average arc origin point.

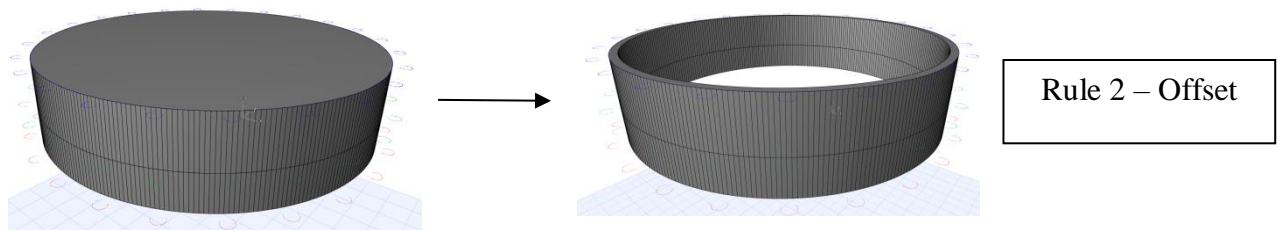


Figure 5.15: Rule Two – Offset an irregular external wall surface by a uniform distance to create a higher level of detail model containing both internal and external wall surfaces.

In order to offset the external building surface, a new polygon is created for each cut-section. The previously created framework of standard radial lines is used to calculate distances from the average arc origin to nodes representing the internal wall surface. This distance is calculated for each standard line on each section using Equation 5.5. Coordinates for the internal points on a wall surface are then calculated using this distance and the angle of each standard line (Equation 5.6). Figure 5.16 shows an example of polygons created for internal wall surfaces using a framework of standard radial lines. For illustrative purposes, Figure 5.16 shows a section made up of four completely different arc segments (top of Figure 5.16) along with a section showing a more realistic shape of a cylindrical drum wall containing deformation (bottom of Figure 5.16). In Figure 5.16 a low resolution of sixteen radial lines is also used for

illustrative purposes. A much higher number of standard radial lines are used for a more accurate estimation of curves.

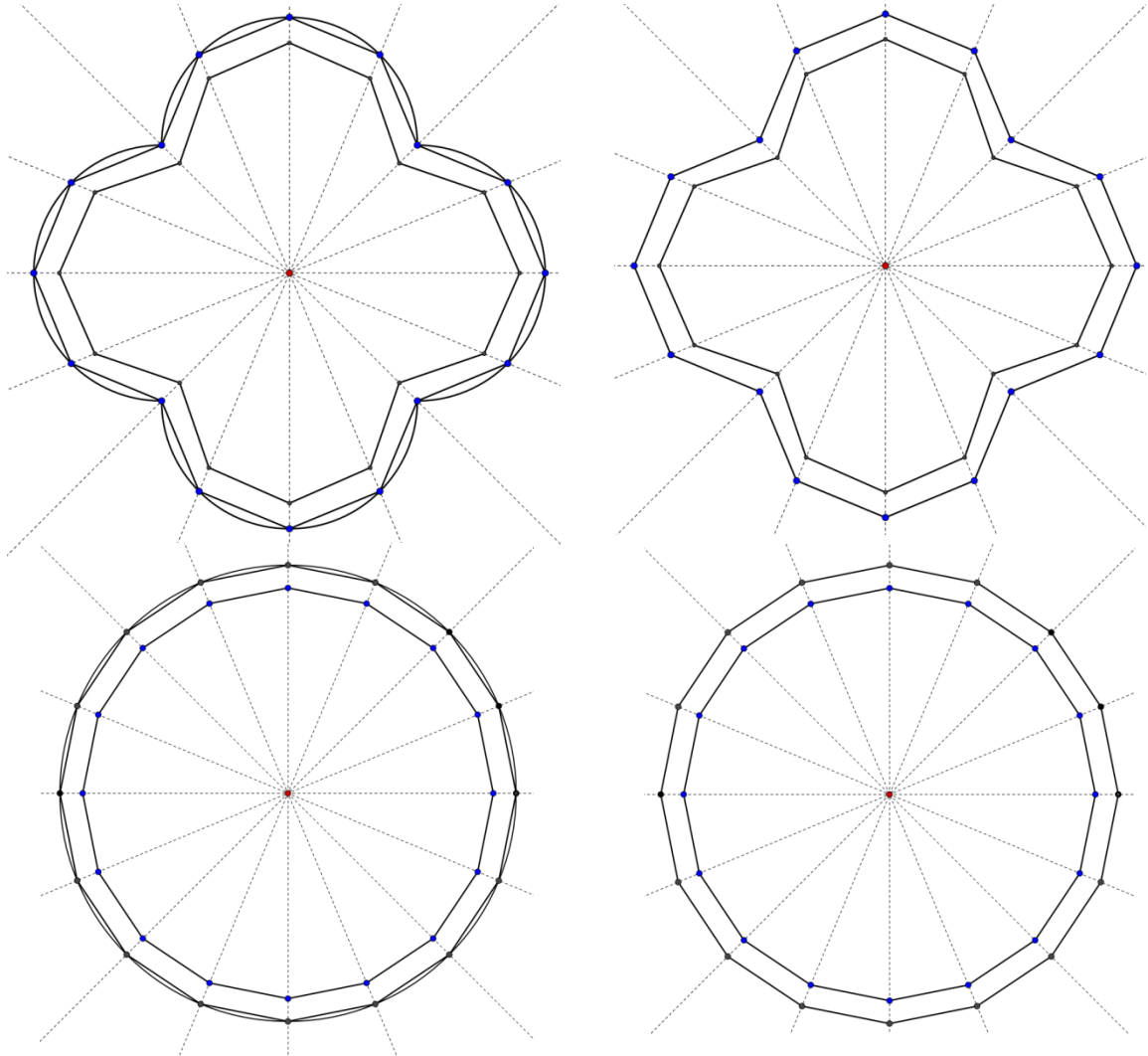


Figure 5.16: Polygon definition for external and internal wall surfaces using a low resolution (16 line segments) to estimate the true arcs.

Finally, the geometry is generated for an internal wall surface by connecting up nodes in different sections using a GDL *RULED* command. Boolean operations are then used to subtract the internal surface from the external surface leaving just the area contained by the wall (Figure 5.15).

$$\text{Distance to Internal node} = \text{distance to external node} - \text{wall thickness}$$

Equation 5.5: Equation to calculate the distance from the average arc origin to an internal node for a standard radial line.

$$\text{InternalNode}_x = \text{averageArcOrigin} + \text{distanceToInternalNode} \times \cos(\text{lineAngle})$$

$$\text{InternalNode}_y = \text{averageArcOrigin} + \text{distanceToInternalNode} \times \sin(\text{LineAngle})$$

Equation 5.6: Equations to calculate the coordinates of an internal node using a distance and angle from a known coordinate.

### 5.3.3 Rule Three: Split Rule

The third procedural rule is a split rule which is used to subdivide a previously created wall surface to create openings. This split rule is used to split a wall surface into both floors and also tiles. This is different to the previous prototype where a repeat rule was used to create different floors. With this prototype, cut-sections are used to define a wall surface for the complete height of a building so a split rule is used instead of a repeat rule to create different floors. The number of floors and tiles on a floor are parameters for this rule.

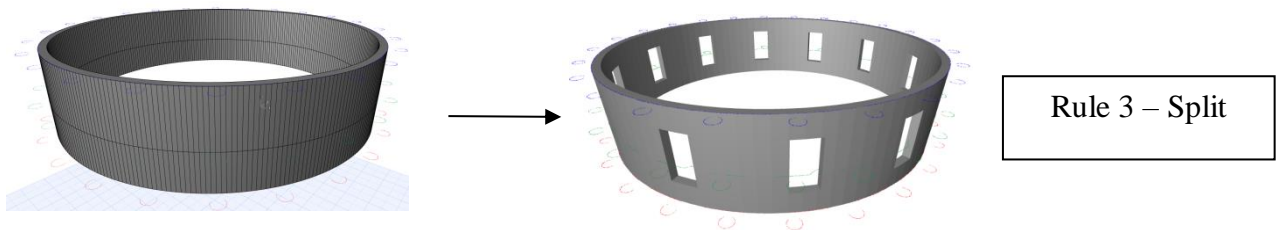


Figure 5.17: Rule Three – Split rule to subdivide an irregular wall surface into floors and tiles containing openings.

The workflow for creating tiles and openings for this prototype is similar to the workflow for the split rule with the previous prototype (Figure 4.21) but involves a number of additional steps and calculations. This is due to the fact that splitting curved wall geometry with the previous prototype involved subdividing a single arc to create tiles and openings. With this prototype, however, a curved surfaced is defined by multiple arcs contained in multiple sections.

Figure 5.18 shows a workflow diagram for the steps involved in the split rule for this prototype. Firstly the wall surface is divided into planar tiles by dividing the angle for the complete wall surface (360 degrees in this case) by the number of tiles (Equation 5.7). These angles (split angles) are then used to calculate split line coordinates on the wall surface. The split line coordinates define the start and end points for planar tiles (Figure 5.19). The split line coordinates along with tile opening coordinates are calculated using a single polygon cut-section. If there are only two cut-sections then the lowest section is used, otherwise, a middle section is chosen to calculate tiles and openings.

For this prototype when calculating the split line coordinates, first the arc number that intersects a split line needs to be found as a section can contain multiple arcs (Figure 5.19). The arc is found using the process shown in Figure 5.12 with the previous rule where split angles are compared to the angles to each arc start and end point. Once the arc is found, the intersection is then calculated between this arc and the line from the average arc origin defined by the split angle (Figure 5.19). This intersection between a line and an arc is calculated using the steps outlined in Table 4.3. After split line coordinates are calculated then planar tile lengths and angle are calculated using Equation 4.2 and Equation 4.3. Next the absolute x and y coordinates for window openings are calculated on a circular wall using the steps in Table 5.2.

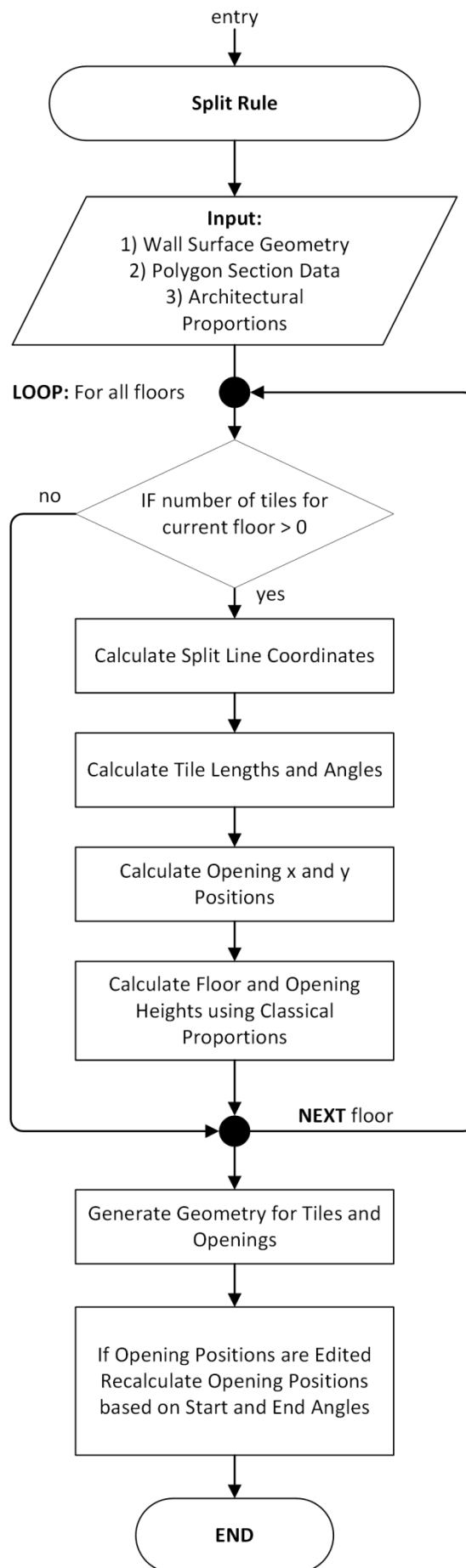


Figure 5.18: Flow diagram showing the steps for rule three –Split Rule.

$$\text{Split Angle} = \frac{360}{\text{Number of Tiles}}$$

Equation 5.7: Formula to calculate split angles for subdividing a circular wall surface face.

Table 5.2: Steps to calculate the x and y coordinates for window openings on an irregular circular wall.

1)	Calculate absolute coordinates of window opening points on planar tiles using Equation 4.7. Windows are centred on a planar tile to ensure resulting window openings will be equally spaced on a circular wall (Figure 5.19).
2)	Calculate coordinates of perpendicular lines from window openings on planar tiles using Equation 4.10 (Figure 5.20).
3)	Calculate intersections of perpendicular lines and curved wall face (arcs) using the steps in Table 4.3. If split line coordinates for the current tile are on different arcs then the arc intersecting each perpendicular line first needs to be found using the process outlined in Figure 5.12 (Figure 5.20).
4)	Calculate the remaining two points on window opening by extending lines from the average arc origin through the first two opening points on an arc (Figure 5.21). Equation 4.3 is used to calculate the angle of these lines. Intersection points between the lines and arcs are then used to calculate the opening coordinates (Table 4.3).

After the x and y coordinates are calculated for openings, next the z coordinates are calculated for window openings. If a wall is split into multiple floors then the heights of each floor are also calculated. The opening formation levels, opening heights and floor heights are all calculated using classical proportions (Figure 3.5) with a set of parametric expression defining parameters. These parametric expressions set the initial values for parameters which can later be graphically edited by a user. After all opening coordinates are calculated then the geometry for these openings can be generated. Boolean operations are again used with opening coordinates to cut window openings from wall geometry.

When a user is interactively editing a window opening in a circular wall, the opening position is edited by changing the start or end angles of the opening from the average arc origin to rotate the opening around the curve (Figure 5.22). The steps previously described for generating a window position, however, are based on a calculation using the linear window width and not the start and end angles of a window. These start and end angles are instead calculated after a window has been created. This means, using the previously described steps to calculate an opening position, it will not be possible to

change a window position based on an opening start or end angles. This requires another method for calculating new window positions for interactive editing. The solution for this involves first finding the arc that intersects the line from the average arc origin with the new start or end angle. Then the intersections are calculated between the arc and line to calculate new openings based on the start or end angles. After the new opening positions are calculated a new window width is also calculated using a distance formula between the inner opening points (Equation 4.2).

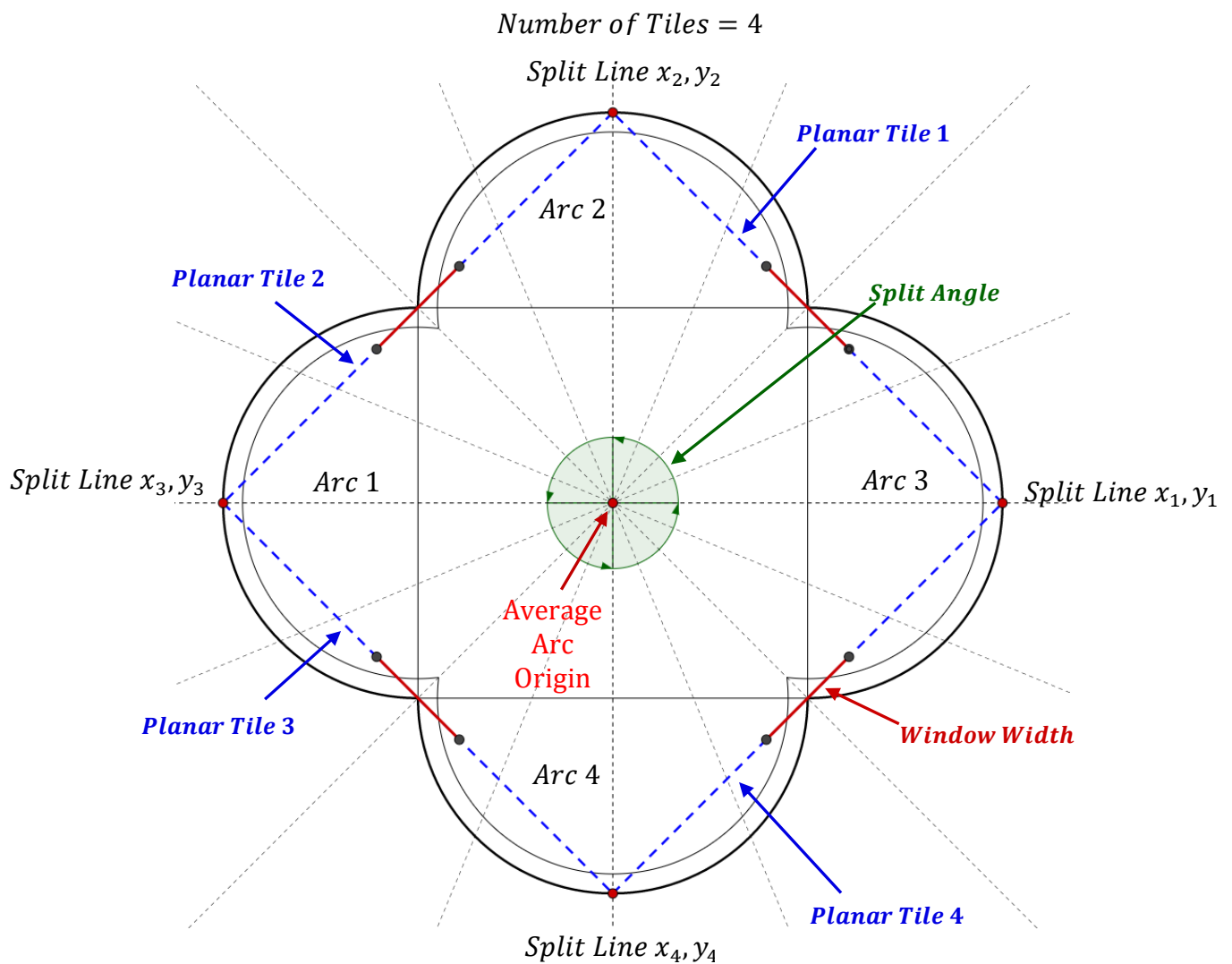


Figure 5.19: Diagram showing the subdivision of a wall surface using split angles to calculate split line  $x$  and  $y$  coordinates.



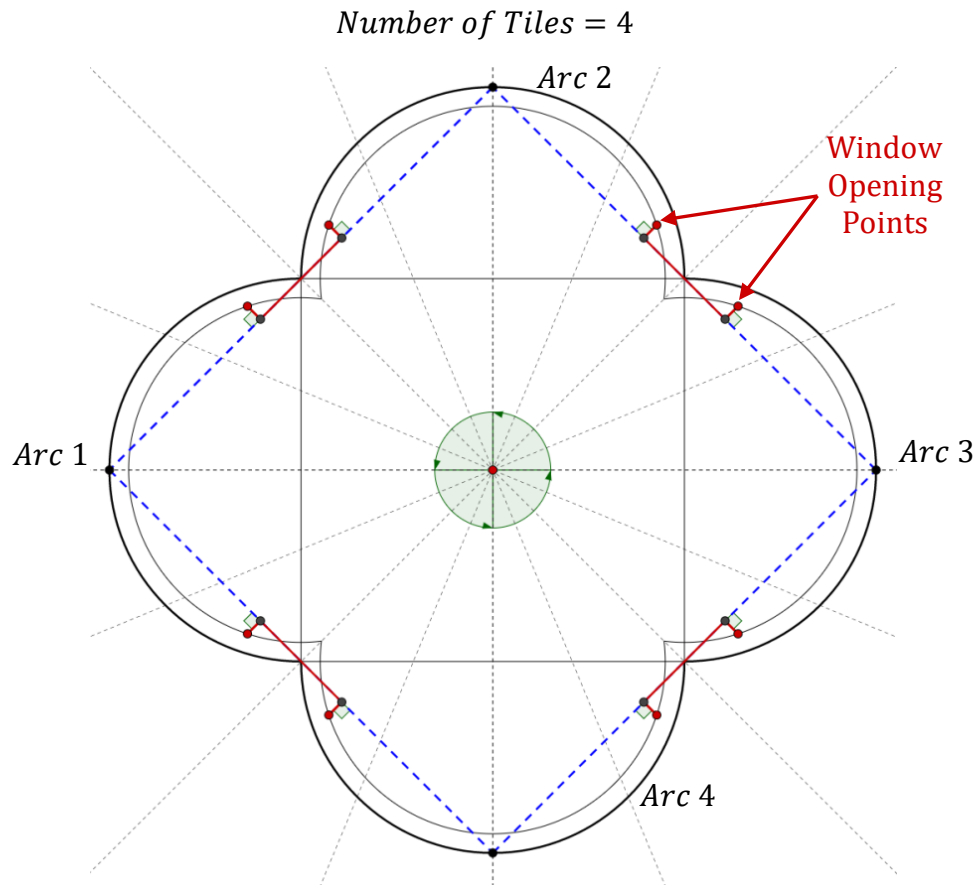


Figure 5.20: Diagram showing solution for calculating the first two window opening points on a subdivided circular wall surface.

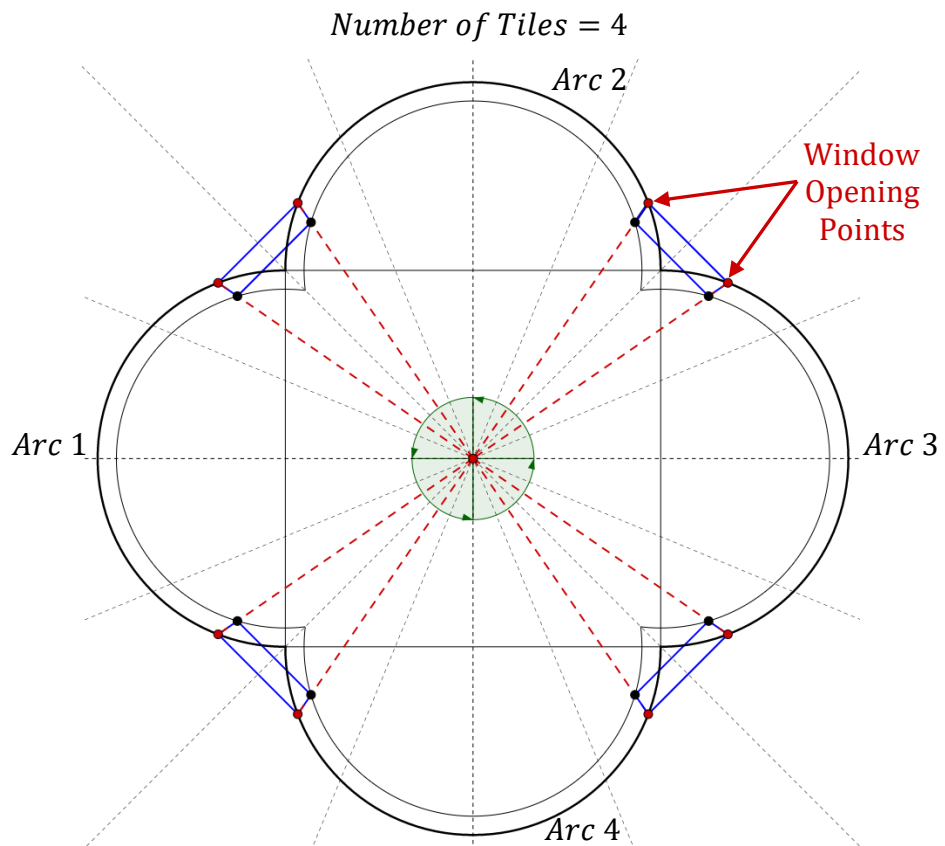


Figure 5.21: Diagram showing solution for calculating the remaining two window opening points on a subdivided circular wall surface.

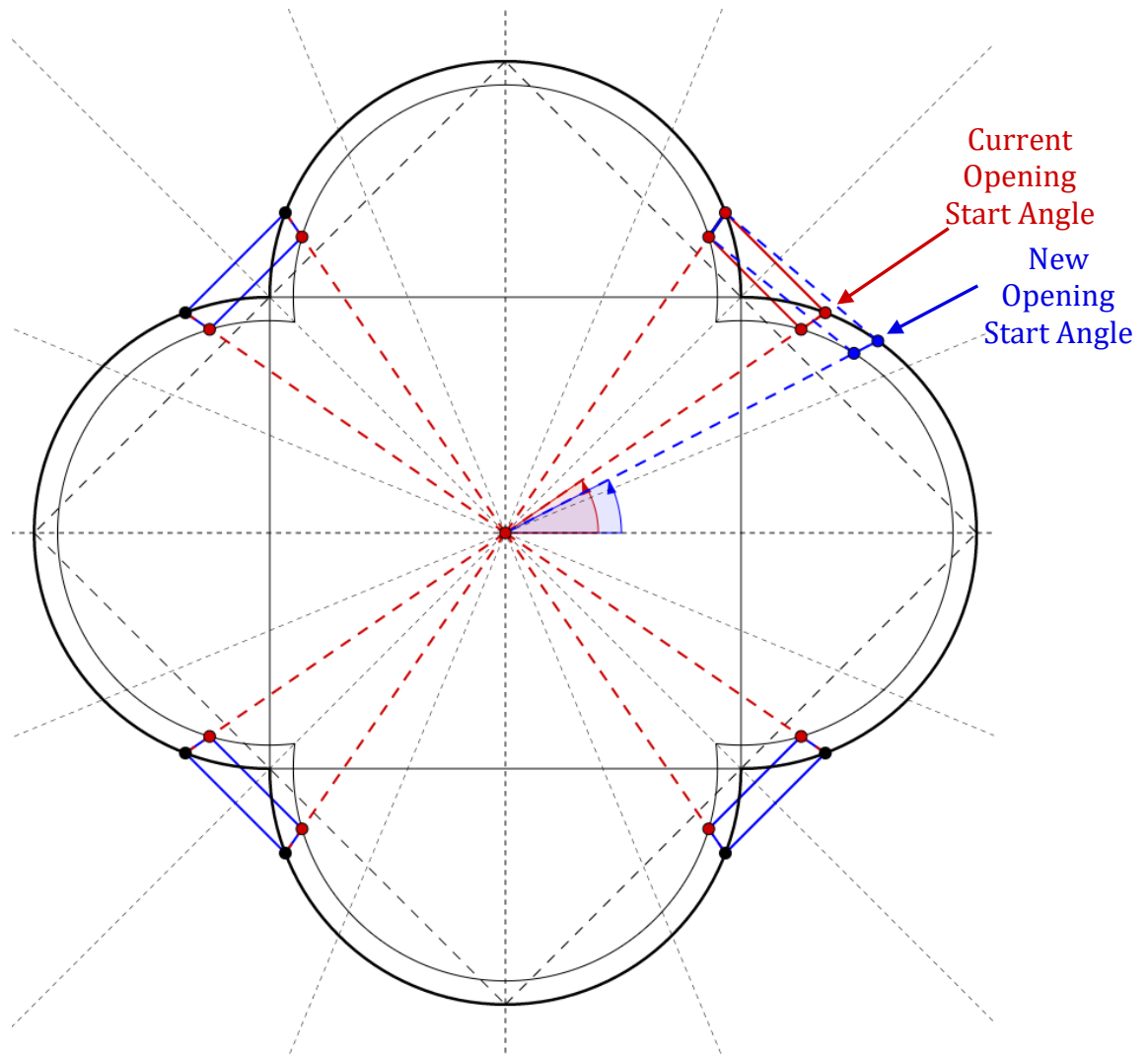


Figure 5.22: Diagram showing current (red) and new position (blue) of window opening after editing a parameter for the window opening start angle.

### 5.3.4 Rule Four: Replacement Rule

The fourth rule for this prototype is a replacement rule that is used to add additional parametric shapes and objects to the generated building model (Figure 5.23).

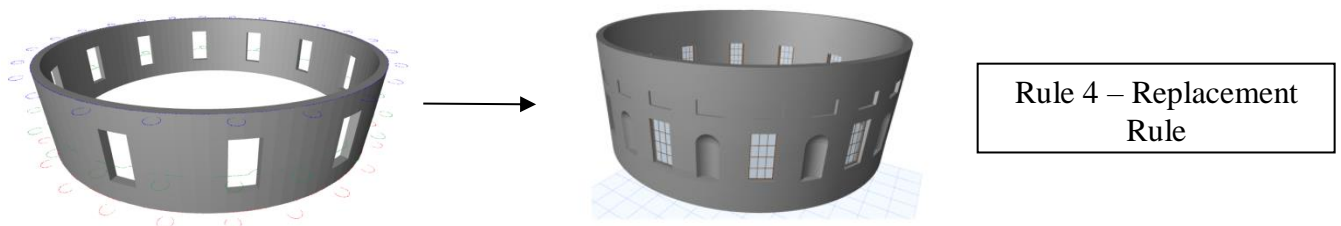


Figure 5.23: Rule Four – Replacement Rule to add new parametric library objects or shapes and replace existing shapes.

The referencing and semantic information attached to tiles allows additional detail and objects to be easily placed anywhere on the generated building or wall structure. Objects can be added to a specific tile, groups of tiles or all tiles using semantic attributes. Figure 5.24 shows an example of parametric windows automatically added to existing wall openings. In this case, a parameter is set to add window objects to all tiles on the generated model. The width, height and location parameters of the window object are automatically matched to the opening parameters that they are placed in. Other parameters of window objects can be set from a dialogue box or by interactive editing in the 2D or 3D windows.

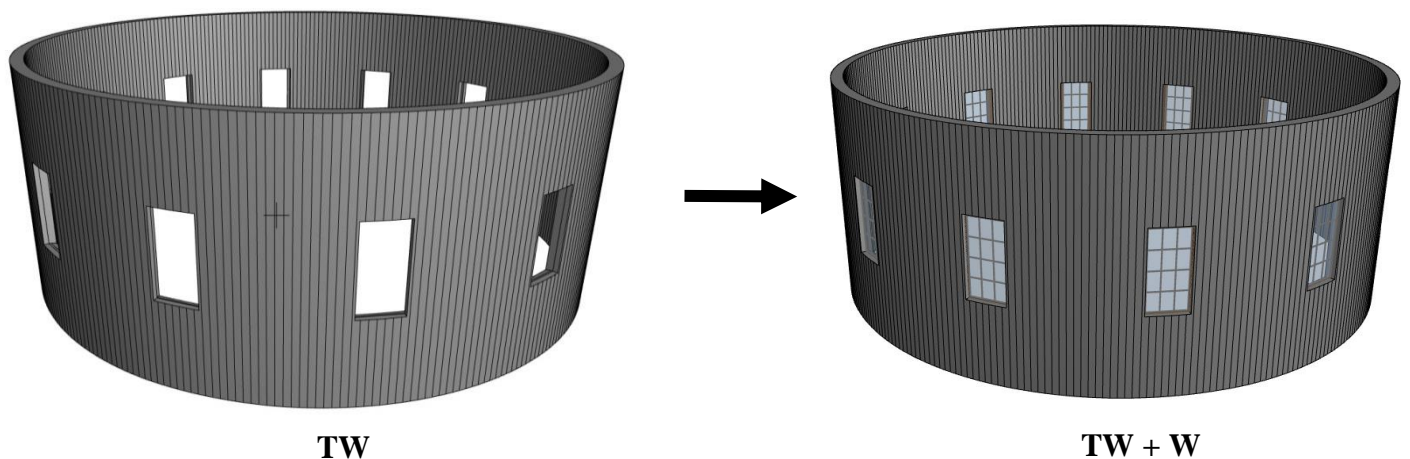


Figure 5.24: Replacement rule used to add parametric windows ( $W$ ) to existing circular wall tiles ( $TW$ ).

Figure 5.25 shows two additional objects added to a procedurally generated model, an arch-top niche and a rectangular niche. Similar to the previous window object, these objects can be added to individual tiles, groups of tiles or all tiles by setting parameters in a dialogue box. With this replacement rule, tiles can also have more than one object placed in them as seen in Figure 5.25, where a rectangular niche is placed in the same tile above window objects and arch-top niches. Tile objects can also be placed by alternating different objects on a floor. Alternating objects are specified in a dialogue

box and the resulting arrangement is generated for the number of tiles set on that floor (Figure 5.25).

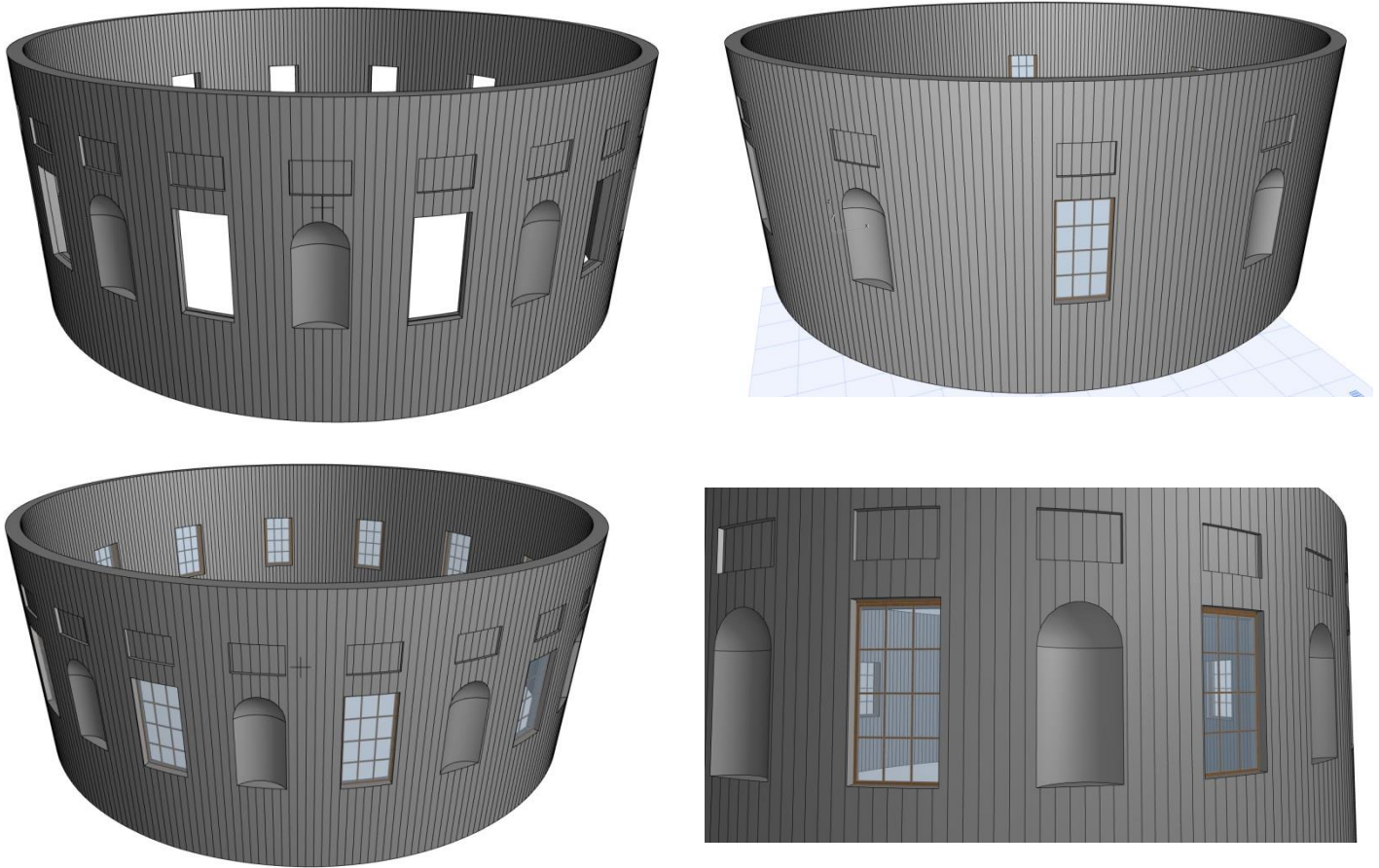


Figure 5.25: Replacement rule used to add additional parametric objects to a generated wall or building model. Arch-top niches, rectangular niches and windows are added using this replacement rule.

Arch-top niche objects are created using a cylinder and sphere primitive shapes which are then cut from a wall using Boolean operations (Figure 5.26). User parameters for an arch-top niche include the start and end angles (used to calculate the width of a niche), niche depth, height and formation level. Rectangular niche objects are created from a 2D polygon outline, positioned at a formation level and extruded to a height of the rectangular niche (Figure 5.27). Boolean operations are again used to cut rectangular niches from a wall surface. Rectangular niches have the same parameters as arch-top

niches, start and end angles (used to calculate the width of a niche), niche depth, height and formation level.

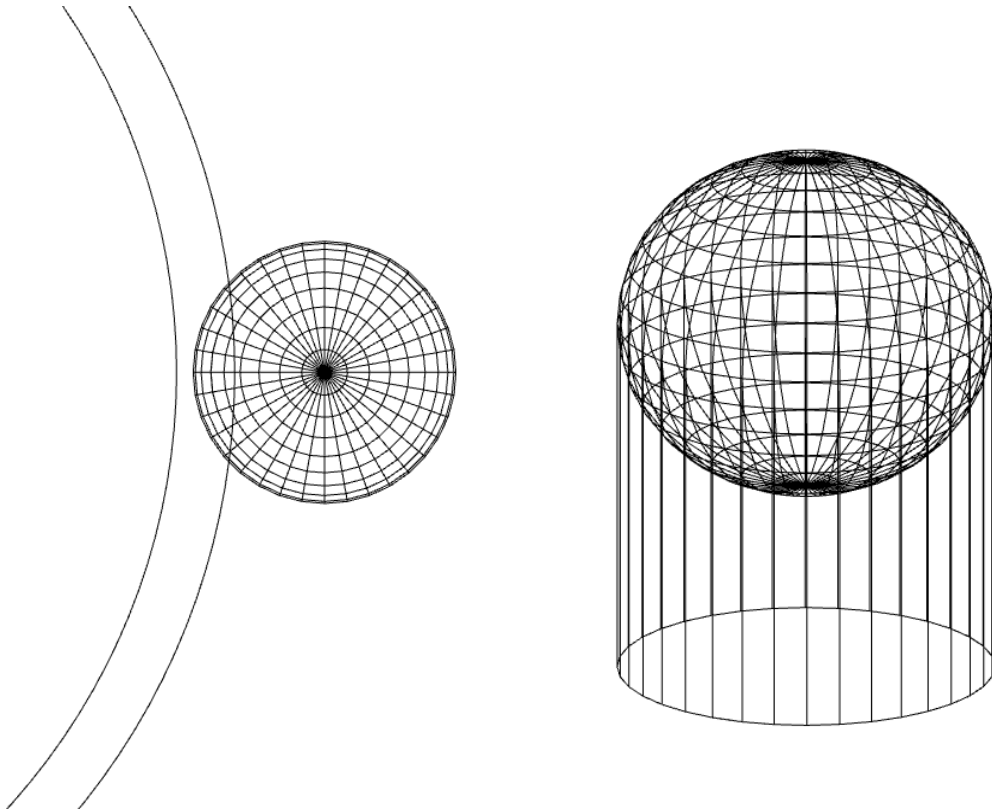


Figure 5.26: Primitive shapes used create arch-top niche objects which are cut from a wall using Boolean Operations.

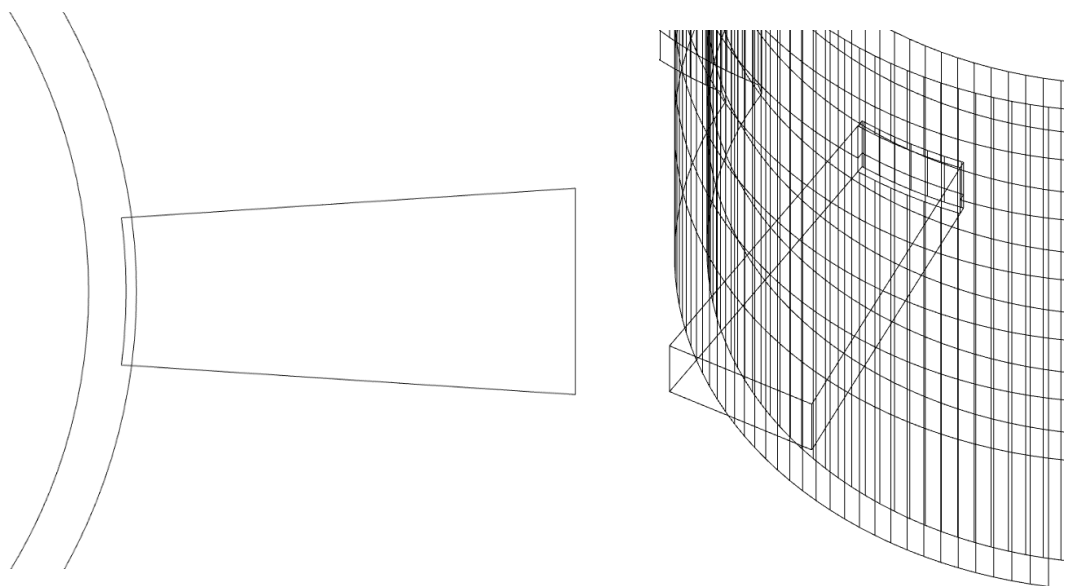


Figure 5.27: Polygon outline used to define a rectangular niche object which is cut from a wall surface using Boolean operations.

The position of niche objects on a wall surface is calculated using the same methods as described previously for calculating a tile opening position (Table 5.2). Niche widths are first calculated on a planar tile and then projected onto the curved wall surface (Figure 5.29). Once the coordinates required for a particular niche object are calculated then the object is placed. When placing an arch-top niche, the centre and radius of the cylinder and sphere are calculated from the coordinates of three points on the arc (niche start and end points and a midpoint defined by the niche depth). This is calculated by finding the intersection of the perpendicular bisectors through chords joining the three points (Figure 5.28). The radius of the sphere and cylinder are then calculated using the distance formula (Equation 4.2). When a rectangular niche is placed the back surface of the niche is defined by the arc of the curved wall surface that it is placed in (Figure 5.29). This requires the arc data defining a curved wall surface to be inputted into the object to assist in the generation of this object. This enables accurate modelling of rectangular niche objects which may contain deformation as dictated by the wall surface that it is placed in.

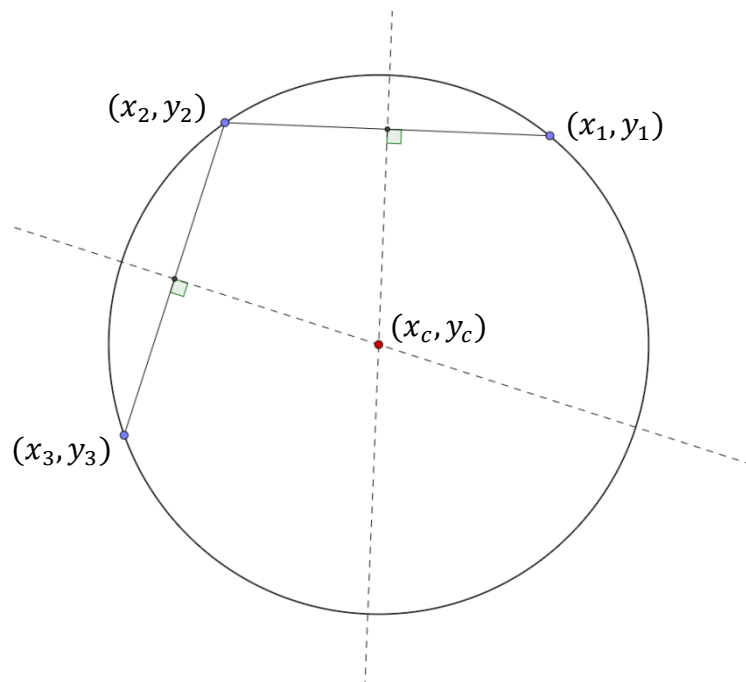


Figure 5.28: Diagram showing solution for calculating an arc centre point using three points on the arc.

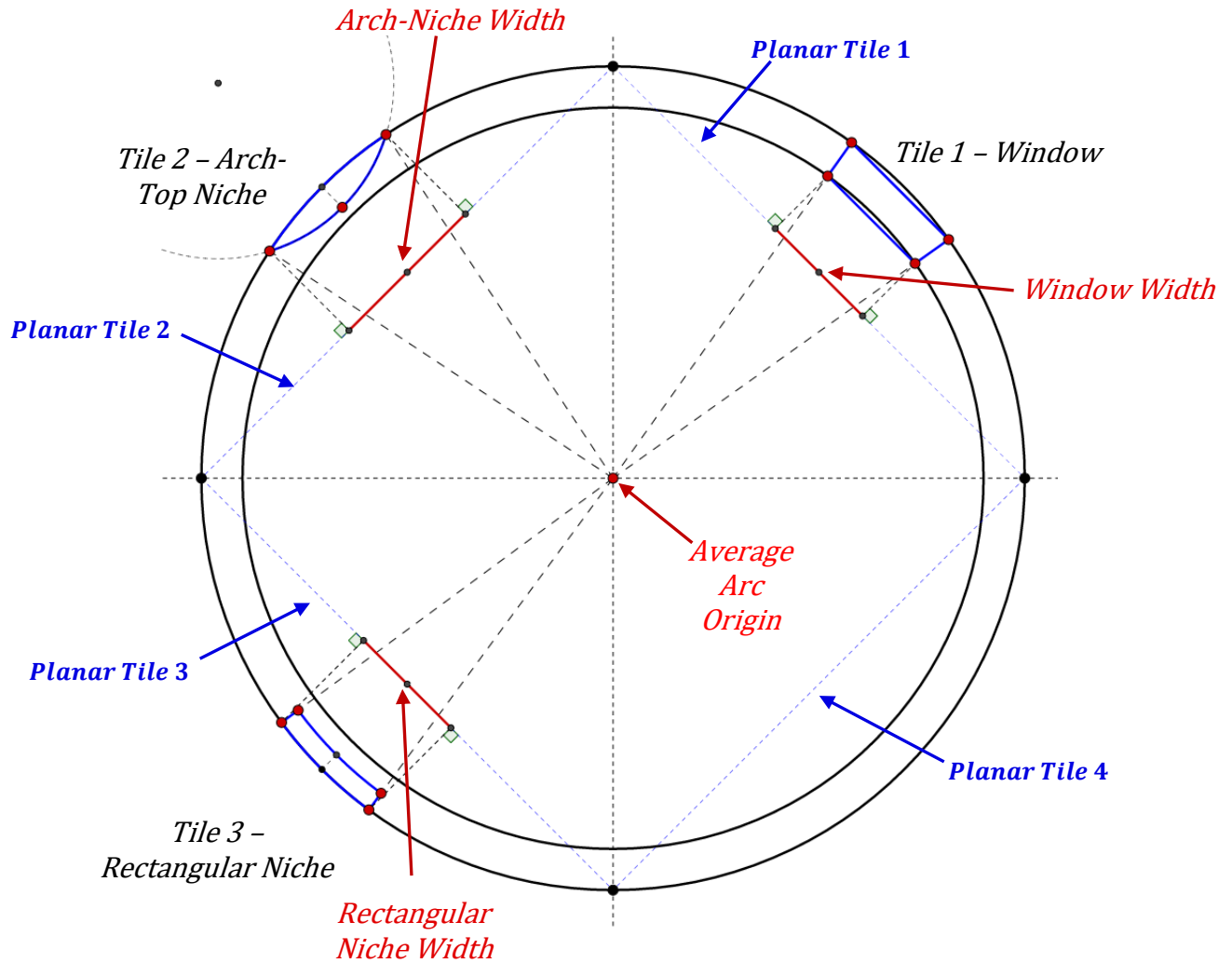


Figure 5.29: Diagram showing the solution for calculating coordinates for window, arch-top niche and rectangular niche objects on an irregular circular wall surface.

Columns can also be added to existing geometry with a replacement rule or alternatively with the first procedural rule for this prototype, “surface generation from cut-sections”. Using a replacement rule, standard column shafts from the HBIM library of objects can be placed around a circular wall surface. Arc data defining a curved wall surface is offset to define the path for the columns which are equally positioned around this offset curve. These columns are based on architectural data and represent ideal columns as originally designed. The second approach allows for modelling column shafts in their true condition which may contain deformation or non-vertical objects. This involves using horizontal cut-sections through a column shaft to define the surface for the



columns (Figure 5.30). Procedural rule one from this prototype is used to generate the surface for a column shaft which connects each cut-section (Figure 5.2). Using this rule the column shafts represent their true condition and are also automatically positioned in their correct position based on the cut-sections provided from survey data. Figure 5.30 shows twenty-four column shafts automatically generated from cut-sections with this rule along with ideal columns positioned with the replacement rule.

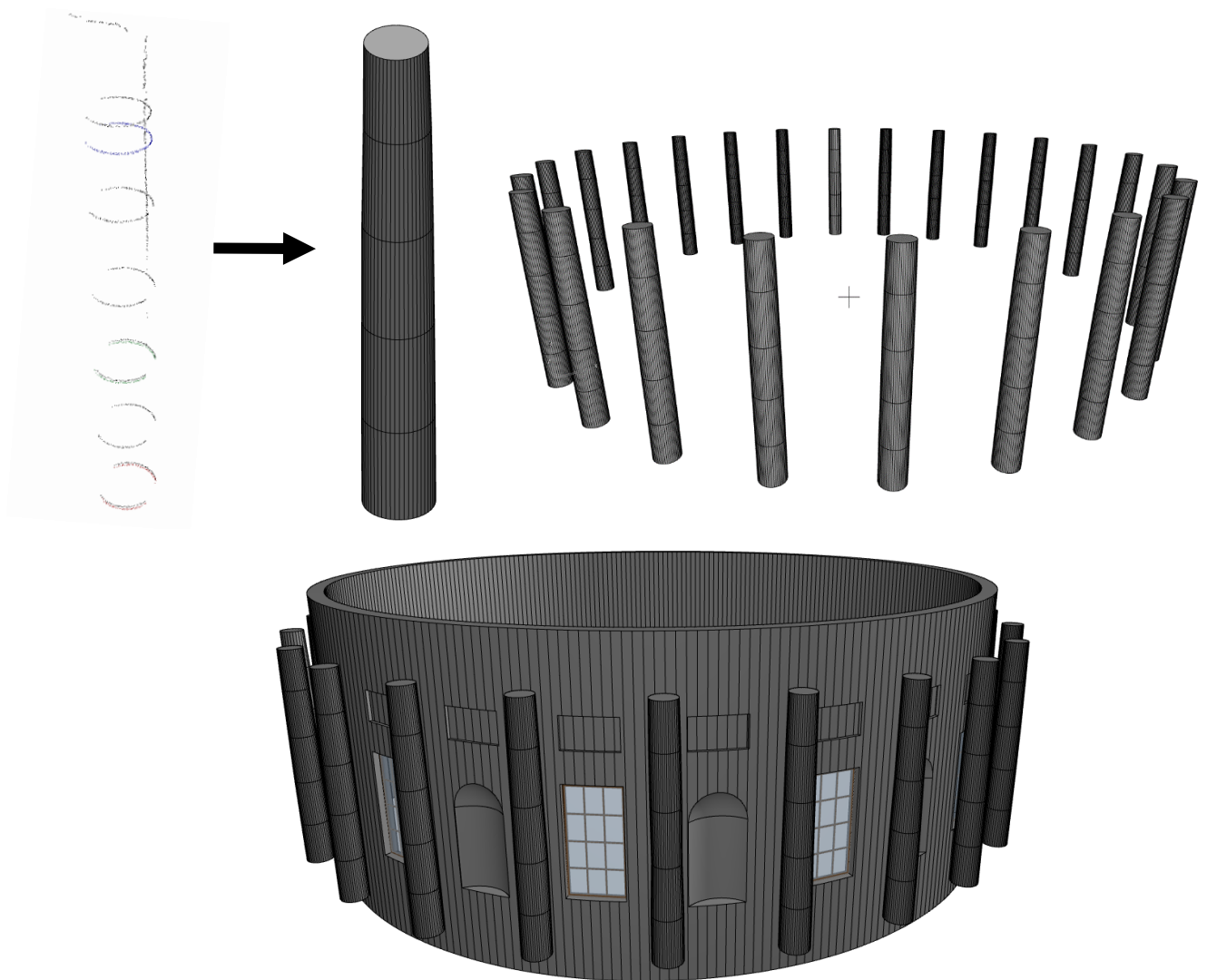


Figure 5.30: Library objects for a column shaft automatically generated from survey data (cut-sections) with procedural rule one (top) or alternatively ideal columns created from architectural data, added to existing geometry with a replacement rule (bottom).



## 5.4 Prototype III: Implementation of Procedural Rules

The procedural rules for prototype III have again been implemented as a plug-in to the ArchiCAD BIM software using the Geometric Descriptive Language (GDL) and the C++ programming language. With GDL, the new procedural rules used to calculate parameters are implemented in a master script, which is executed before any other scripts. The vocabulary shapes or library objects are coded in a 3D script and stored in separate subroutines. An executive script at the top of the 3D script is used to call subroutines and apply coordinate transformations as required.

The first procedural rule for this prototype is implemented with both GDL and C++ to retrieve selected polygon sections within the ArchiCAD environment. This polygon section data is then passed to the GDL object for use with subsequent procedural modelling rules. The C++ functions used to for this prototype are shown in Table 5.3. The main function “*Do\_editLibraryObjectSection*” is used to retrieve the polygon data for each section and modify the GDL object based on these parameters (Table 5.4). Within ArchiCAD, this function is accessed from the new HBIM menu (Figure 5.31). Using GDL, the remaining calculations for the first rule are coded in a master script and the geometry is then generated in the 3D script using a *RULED* command (Figure 5.32).

Table 5.3: C++ Functions for implementing rule one – Procedural Surface Generation from Cut-Sections

<b>Function 1:</b>	<code>Do_editLibraryObjectSection();</code>
<b>Function 2:</b>	<code>GetSelectedElementPolygon();</code>
<b>Function 3:</b>	<code>CheckEnvironment();</code>
<b>Function 4:</b>	<code>RegisterInterface();</code>
<b>Function 5:</b>	<code>Initialize();</code>
<b>Function 6:</b>	<code>FreeData();</code>

Table 5.4: Steps for function one “Do\_editLibraryObjectSection” shown in Table 5.3.

1) Retrieve and store selected polygon section data by calling function three “GetSelectedElementPolygon”.
2) Retrieve the global unique identifier (GUID) for the library part resource which is to be edited.
3) Change the default parameters of the library part resource. This updates parameters of the GDL file with parameters of the selected polygon sections.
4) Create an instance of the library object in the database and place on floorplan.

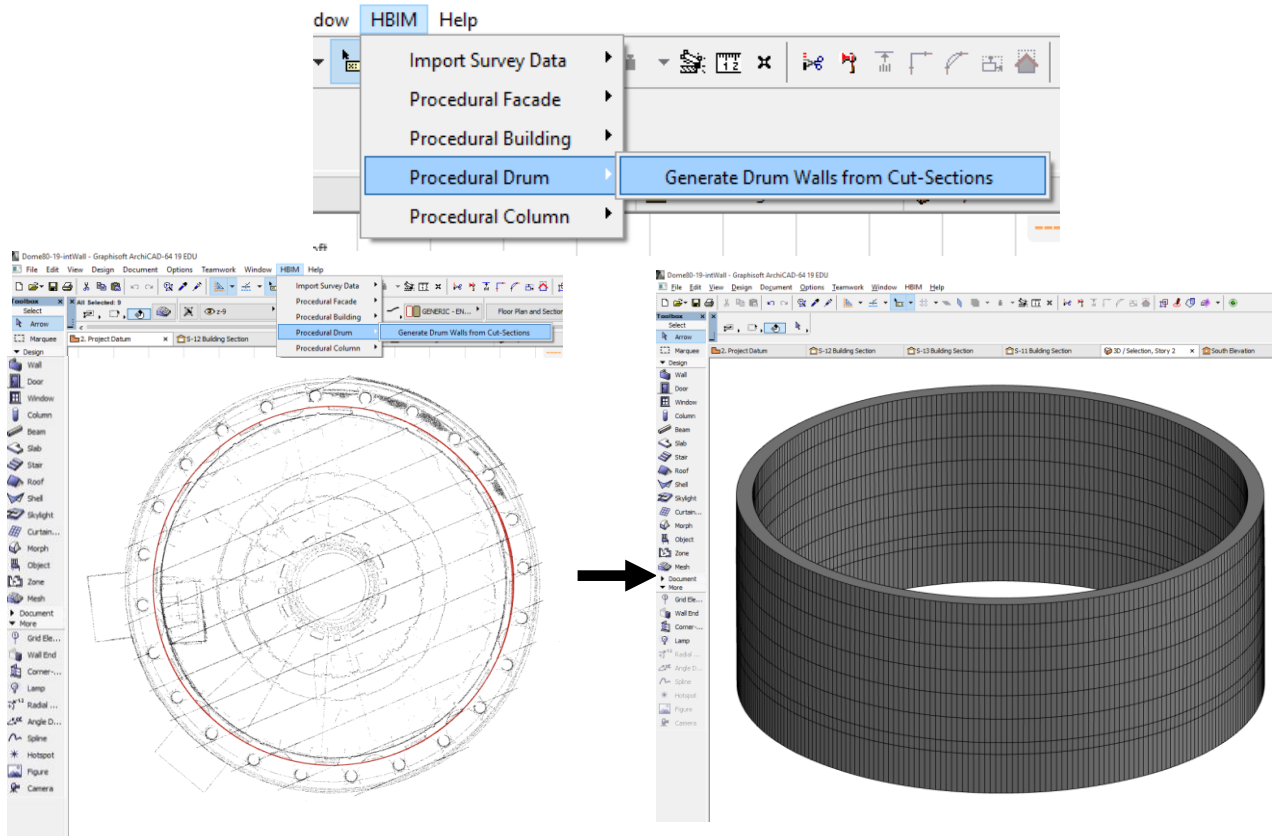


Figure 5.31: New HBIM menu to access functions of prototype plug-in (top). Nine polygon sections are selected (bottom left) which are used to generate the non-vertical wall containing deformation (bottom right).

The subsequent rules which include the offset rule, split rule and replacement rule are implemented in a similar way, with procedural rules that perform calculations implemented in a master script and geometry generated from a 3D script.

```

30 ! =====
31 ! Executive Script:
32 ! =====
33 !-----
34 !!!External Wall Surface & Hotspots:
35 !-----
36 GROUP "ExternalWall"
37 !!!External Wall Surface:
38 FOR section = 1 to (nSections - 1)
39     ADDz sectionHeight[section]
40     FOR lineNumber = 1 TO nNodes
41         PUT section_x[lineNumber][section], section_y[lineNumber][section], 0
42     NEXT lineNumber
43
44     FOR lineNumber = 1 to nNodes
45         PUT section_x[lineNumber][section + 1], section_y[lineNumber][section + 1], sectionHeight[section + 1] - sectionHeight[section]
46     NEXT lineNumber
47
48     RULED nNodes, 1+2+4+16+32, GET (NSP)
49
50     DEL 1
51
52 NEXT section

```

Figure 5.32: Sample GDL code from the 3D script where a *RULED* command is used to generate the geometry for an external wall surface from any number of sections.

Methods for optimising the speed of this prototype have also been implemented. Due to the large number of calculations that are required for different rules, it is crucial that only the required calculations are carried out when a script or rule is executed. This is especially important while a model is being updated during interactive editing. When interactively editing a model, users expect a model to be fully responsive and update instantly. In some cases, however, many calculations need to be performed to calculate new parameters after an edit has been made. To ensure fast editing, the main calculations are initially run in sequence when a model is first generated from section data and a minimal amount of calculations are performed as required when an edit is being made.

With GDL the complete master script and 3D script are executed when a parameter is edited. For this reason, it is important to control when different parts of these scripts need to be executed. Using a global parameter in GDL, “*GLOB\_MODPAR\_NAME*”, it is possible to identify what parameter is being edited when a change is made to a model.

This is very useful to control and limit calculations based on what parameters are being edited. Unlike the previous two prototypes, when performing edits with this prototype on objects such as windows or niche objects, a lot of calculations need to be performed. This is because editing the width of an object cannot be carried out directly on a curved surface. Instead, the start or end angles of an object are edited which then calculate the new coordinates on the curved surface and the new width. As wall surfaces with this prototype can contain multiple arcs in a section, calculations involve first finding the arc that the new endpoints of an object will intersect and then finding the intersection points to calculate the new coordinates and width of the object.

With this prototype, it is possible to edit all objects in a procedurally generated model simultaneously. This requires many calculations so it results in slightly longer times to update. Editing an individual object, however, should be capable of updating instantly. To achieve this, the global GDL parameter, “*GLOB\_MODPAR\_NAME*” is used to identify which object type is being edited (window, arch-top niche or rectangular niche) and only apply the calculations to tiles containing these objects. A limitation of this “*GLOB\_MODPAR\_NAME*” function, however, is that it can only return the parameter name that is being edited and not a specific variable in an array. This means that if one window object is being edited the calculations will need to be performed on all tiles containing window objects in a model, as opposed the individual tile being edited. This is because an array stores the results of a parameter for all instances of that objects in the one array and the specific window being edited cannot be identified within the array despite the referencing.

In order to enable faster editing, a solution was developed to overcome this problem and to enable a specific tile object to be identified when edited. This requires storing a copy of the values of certain parameters in arrays every time a change is made to a specific

tile object. By comparing the values in the copied and main arrays it is possible to identify which value has changed as only the value of the object being edited will be different in both the main array and copied array. This enables the specific tile object being edited to be identified from all instances stored in an array. When a specific tile object such as a specific window start angle is being edited, it is then possible to identify that instance within an array and limit the calculations required for that specific tile on that specific floor. For large models with many tiles and floors, this will result in much faster and more responsive editing.

Figure 5.33 shows an example of different models generated with the procedural rules for this prototype. Any arrangement of objects can be generated for any number of floors. Each object is also highly parametric and can be altered in groups or individually to enable efficient mapping to survey data. The process for mapping generated geometry to survey data is discussed in chapter 6.

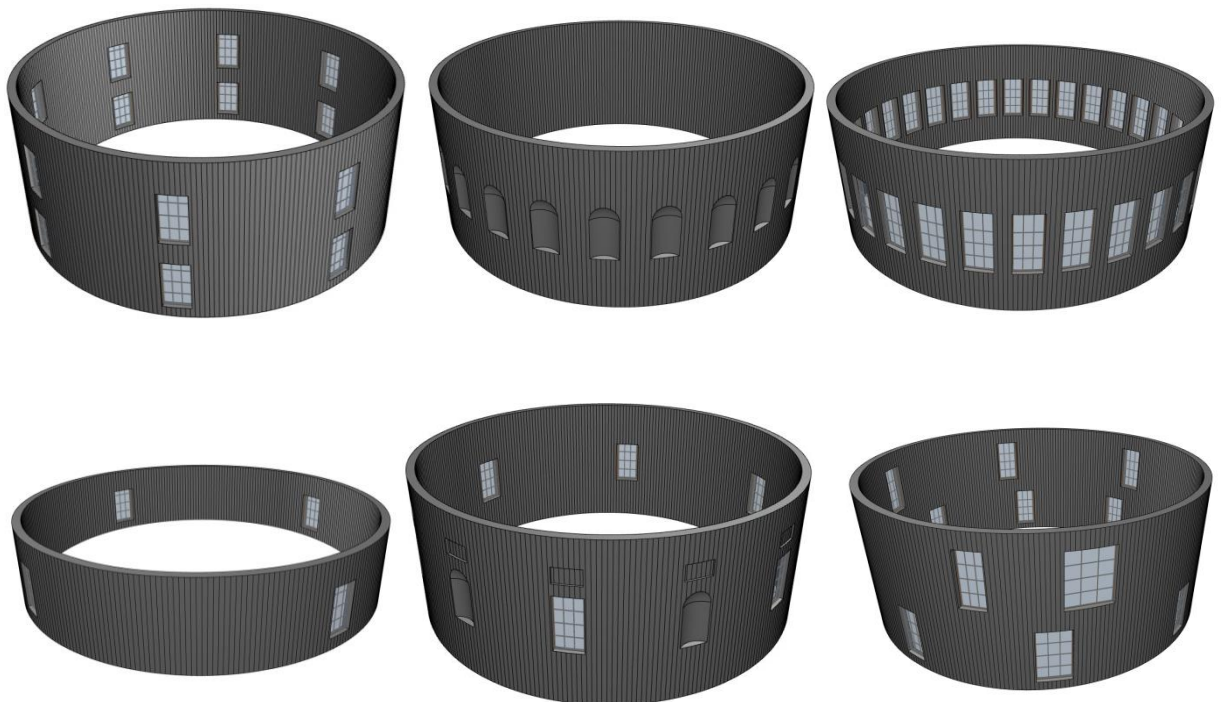


Figure 5.33: Various models automatically generated from cut-sections with the rules and algorithms for the Irregular Procedural Building prototype III.

Figure 5.34 shows an example of highly irregular surfaces generated from cut-sections using the surface generation rule. This shows the capabilities of the prototype for generating non-vertical wall surfaces. The accuracy of a surface generated with this prototype is evaluated in chapter 8. An example of this prototype used in a real case study is also shown in chapter 7.

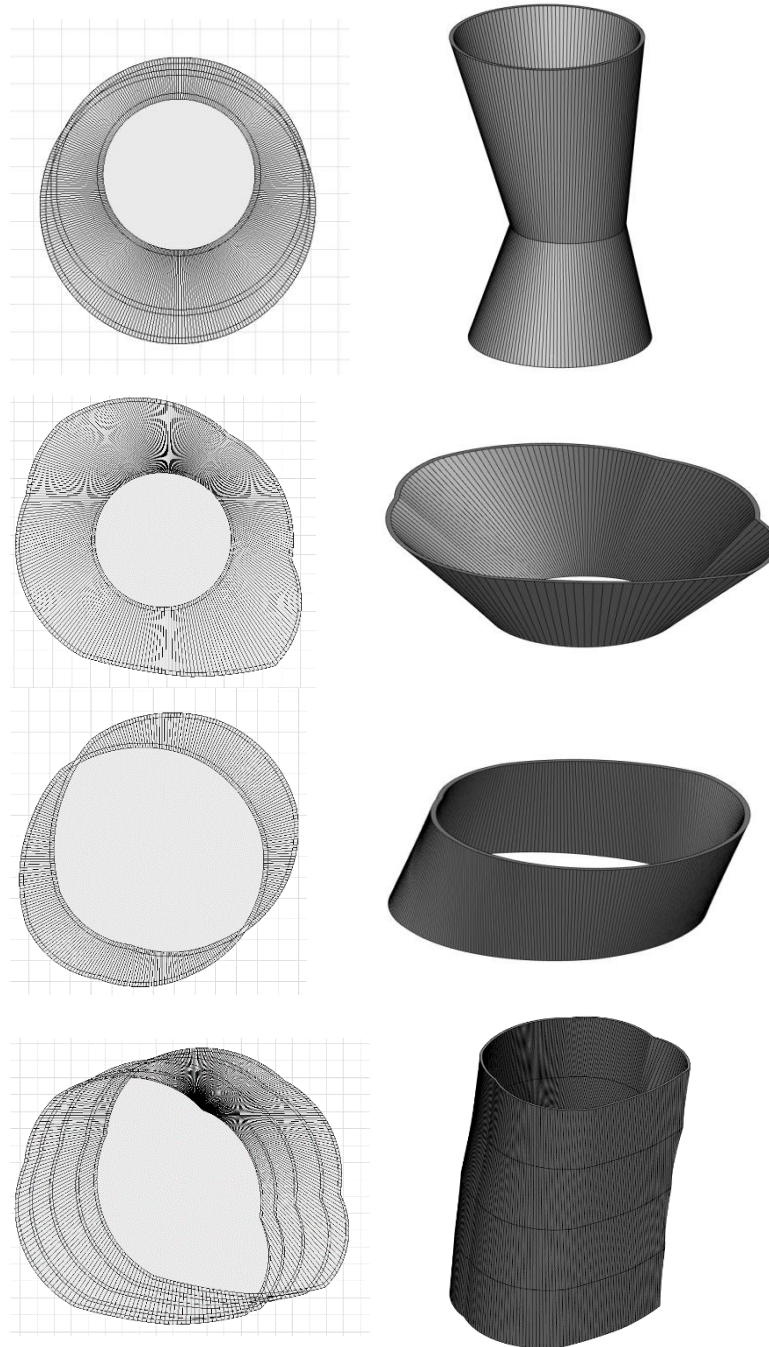


Figure 5.34: Various surfaces automatically generated from cut-sections with the surface generation rule for the Irregular Procedural Building prototype III.

A video showing a demonstration of the new irregular procedural building prototype can be seen from the link below.

**<https://youtu.be/vgla9Aqz0j0>**

## **5.5 Prototype III: Summary**

The irregular procedural building prototype III further developed concepts from the first two prototypes by providing capabilities for accurately modelling deformation in building geometry. With this procedural building prototype, non-vertical circular wall geometry can be generated from any number of horizontal cut-sections. A new set of rules and algorithms have been designed and implemented to:

1. Generate non-vertical surfaces from cut-sections.
2. Offset an irregular surface to convert it to a higher level-of-detail model represented by walls with a uniform thickness.
3. Split a floor or building model into tiles containing openings.
4. Replace and add new vocabulary shapes to previously generated geometry.

Classical architectural rules and proportions also assist with the generation process by providing an initial estimate for the size and positioning of objects. These new rules and algorithms have been implemented as a plug-in to the ArchiCAD BIM software using the Geometric Descriptive Language (GDL) and the C++ programming language with an Application Programming Interface (API). These new rules and algorithms for the irregular procedural building prototype enable the modelling of existing building geometry with a semi-automatic process where the required geometry is first generated and then manually refined to match to specific survey data. The mapping of generated geometry to survey data is described in Chapter 6.

## **Chapter Six: Procedural HBIM – Mapping to Survey Data**

### **6.1 Introduction**

This chapter outlines the process of using the developed procedural modelling prototypes with survey data to model existing buildings. Firstly the steps and issues involved with integrating survey data into a BIM environment are explained. Most BIM software has limited tools for accurately importing and integrating different survey datasets. For this reason, new tools have been developed to enable survey datasets to be precisely positioned in a BIM environment using a common coordinate reference system (CRS). These new tools are described in Section 6.2.1 and Section 6.2.2. Once all survey data is integrated and correctly positioned within BIM software the newly developed prototypes can be used to accurately model building geometry with a semi-automatic procedure. This semi-automatic process is described in section 6.3.

### **6.2 Integration of Survey Data in a BIM Environment**

As discussed in Section 2.3, point clouds obtained from laser scanning or photogrammetry can be used to generate accurate building information models of existing buildings. However, modelling directly to large point clouds can be difficult as large point clouds can be very processor intensive and BIM software is not always capable of handling such large datasets. There can also be accuracy issues when modelling directly to large point clouds in 3D space as it can be difficult to locate the exact position of an object within a dense 3D point cloud. Instead of modelling directly to a complete point cloud, segmented point clouds, orthographic images in elevation and plan, and 2D sections through a point cloud can be used to generate as-built/as-is BIM models.



When metric data is transferred from pre-processing survey software to BIM software, it is crucial that there is no loss in accuracy. One of the biggest sources of errors when integrating different survey datasets in a BIM environment is due to misalignment of different survey datasets. This is caused when survey data is not correctly positioned relative to each other, such as a misalignment between an orthographic image and segmented point cloud. It is crucial that all datasets have been accurately surveyed or aligned into a common coordinate reference system to ensure that there is no loss in accuracy when combining different datasets. This allows different datasets to be precisely positioned in BIM software using this common coordinate reference system as opposed to manually aligning datasets using common points or measurements between points.

As BIM is used primarily for new buildings, there are limited tools in most software for accurately importing survey data using source coordinate reference system. In the ArchiCAD BIM software, for example, there are currently no tools for importing geo-referenced orthographic images in their correct 3D position. A geo-referenced image is an image that is defined in 3D space by the x, y and z coordinates of the image corners. A geo-referenced image contains two files, the image and an accompanying text file with the image coordinates and other metadata. When an orthographic image is imported into the ArchiCAD BIM software it is positioned in an arbitrary 2D position in plan or elevation. This makes it very difficult to precisely position the image relative to other survey data. To overcome this problem a new tool has been developed for the ArchiCAD BIM software to allow geo-referenced orthographic images to be correctly positioned in their correct 3D space using the datasets source coordinates (Figure 6.1).

A new tool has also been developed to overcome this issue and enable 3D points to be accurately imported into any version of ArchiCAD. These new tools for accurately

importing orthographic images and 3D points are described in Section 6.2.1 and Section 6.2.2.

### 6.2.1 New Tool for Importing Orthographic Images to ArchiCAD BIM Software

In order to import geo-referenced orthographic imagery into their correct 3D position, a new tool was created using the Geometric Description Language (GDL). Using this embedded programming language for the ArchiCAD BIM software, a parametric 3D plane was developed to enable the precise positioning of orthographic images in 3D. This new tool allows users to enter the x, y and z coordinates for the four image corners as parameters to generate a 3D plane for the image in its true position. A user can then select the image from a file which is automatically mapped to the 3D plane as texture. Using correctly positioned 3D orthographic images ensures objects mapped to multiple images will be in their correct relative and absolute positions (Figure 6.1). Sample GDL code for this new tool can be seen in Figure 6.2.

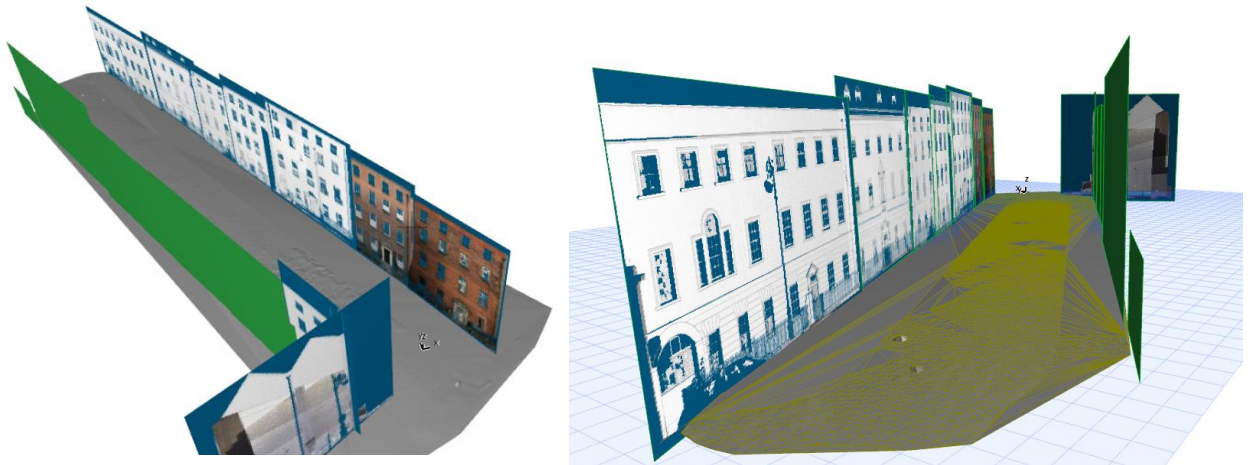


Figure 6.1: Geo-referenced orthographic images imported into ArchiCAD BIM software in their correct 3D position.

```

1  !3D Script
2
3  unID=1 !Unique ID for 3D Hotspots
4  MATERIAL wmat
5
6  PLANE 4,
7  tlx,tly,tlz,
8  trx,try,trz,
9  brx,bry,brz,
10 blx,bly,blz
11
12 !3D Hotspots:
13 HOTSPOT tlx,tly,tlz,unID,A,1 : unID=unID+1 !Base Hotspot
14 HOTSPOT trx,try,trz,unID,A,1 : unID=unID+1 !Base Hotspot
15 HOTSPOT brx,bry,brz,unID,A,1 : unID=unID+1 !Base Hotspot
16 HOTSPOT blx,bly,blz,unID,A,1 : unID=unID+1 !Base Hotspot
17
18 !2D Script
19
20 PROJECT2 3,270,2
21 unID=1 !Unique ID for 2D Hotspots
22
23 !2D Hotspots
24 HOTSPOT2 tlx,tly,unID,A,1 : unID=unID+1 !Base Hotspot
25 HOTSPOT2 trx,try,unID,A,1 : unID=unID+1 !Base Hotspot
26 HOTSPOT2 brx,bry,unID,A,1 : unID=unID+1 !Base Hotspot
27 HOTSPOT2 blx,bly,unID,A,1 : unID=unID+1 !Base Hotspot
28
29
30 VALUES "unit" "Metres","Centimetres","Millimetres"
31
32 IF unit="Metres" THEN
33

```

Figure 6.2: Sample GDL code for new tool to import geo-referenced orthographic imagery.

## 6.2.2 New Tool for Importing 3D Points to ArchiCAD BIM Software

The second new tool has been developed to accurately import 3D points into ArchiCAD which was previously not possible before ArchiCAD version 19. This new tool was again developed with the Geometric Description Language (GDL). A 3D point can be defined with GDL using a *HOTSPOT* command which displays a point based on an x, y and z coordinate. However, after testing this command with a segmented point cloud, it was found to be unsuitable for displaying a large number of points. This is because hotspots firstly only display when an object is selected. This means that a segmented point cloud or cut-section will not display in the 3D or 2D window until it is selected, which makes it unsuitable for its intended use. Secondly, the display size of a hotspot point is quite large and cannot be changed. This means it is very difficult to see individual points that are close together within a large point file. A better method of displaying a large number of 3D points involved representing each point by a very small line segment. Unlike 3D hotspots, line segments display without being selected. The size of a very small line segment is also smaller than the size of a 3D hotspot which

makes it easier to display a lot of points that are close together. Within GDL, a line segment is defined in 3D by an x, y and z coordinate of each line endpoint. In order to represent 3D points with line segments, two coordinates are created for each 3D point. One of the new coordinate points is created by adding a millimetre to each original 3D point and the second new point is created by subtracting a millimetre from each original 3D point. These new coordinate points are then used to define a line segment with the actual 3D point located at the centre of each line. This results in 3D points being represented by very small line segments around 2mm in length (Figure 6.3). An option is also available for a user to display hotspots at the centre of each line segment to enable accurate snapping to original 3D points. Figure 6.4 shows an example of cut-sections and segmented point clouds imported into ArchiCAD with this new tool. Similar to the procedural modelling prototypes, the new tools for importing survey data are also accessed from a new HBIM menu as shown in Figure 6.5.

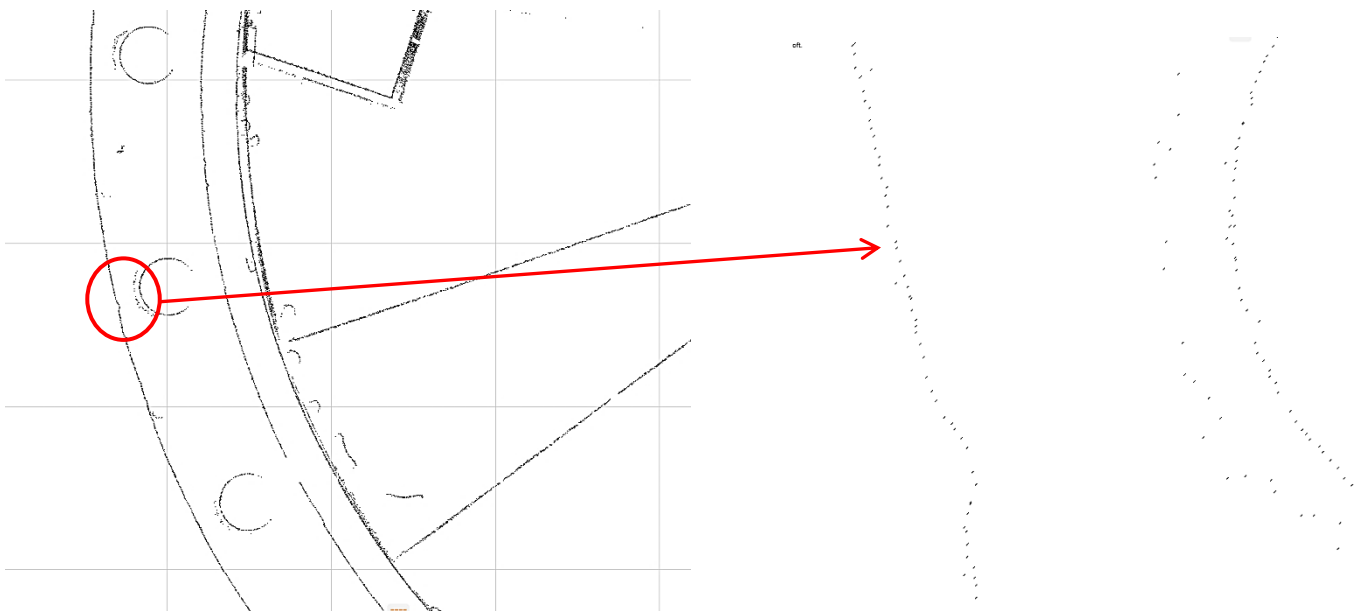


Figure 6.3: 3D points represented with GDL using very small line segments.

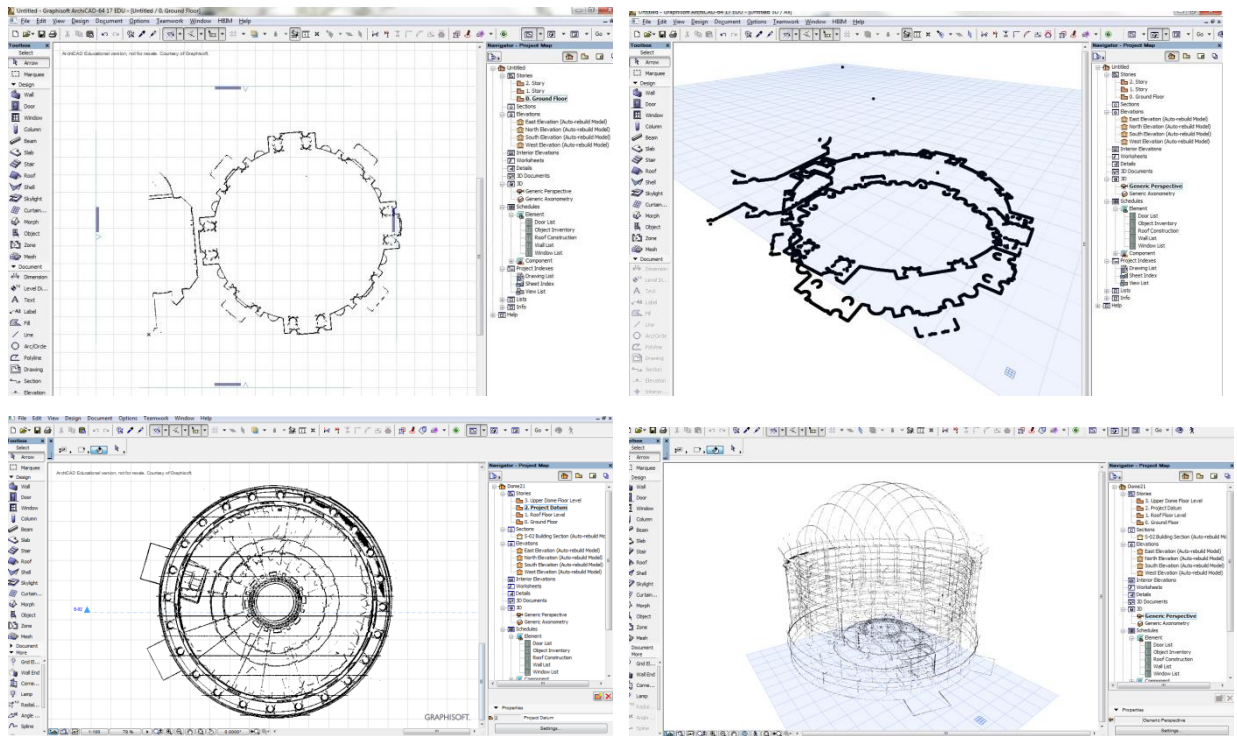


Figure 6.4: Cut-sections and segmented point clouds imported to ArchiCAD BIM software in correct 3D position.

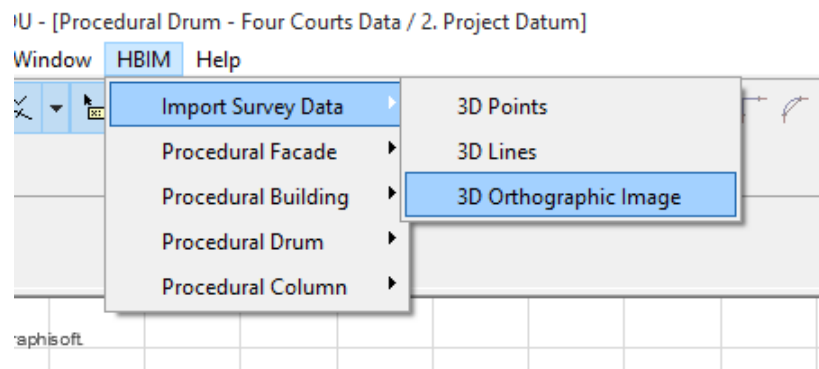


Figure 6.5: Menu command for accessing new tools to accurately import survey data into ArchiCAD BIM software.

### 6.3 Mapping to Survey Data – Prototype I

After all survey data is imported and correctly positioned in BIM software the newly developed procedural modelling prototypes can be used for fast and efficient modelling of existing buildings. When a façade is generated with the first procedural façade prototype, it is first positioned in plan relative to survey data such as segmented point

clouds or orthographic imagery in plan (created from a point cloud) (Figure 6.6). The initial configuration of a procedurally generated façade has two floors and two horizontal tiles as seen in Figure 6.7. This initial configuration contains a door tile (*TD*) and window tile (*TW*) on the ground floor and two window tiles on the first floor. The first stage of mapping this procedural façade template to survey data is to apply the classical proportions to the façade model which provides an initial estimate for the position and size of façade elements. In order to apply these classical proportions, two measurements need to be taken from the survey data and entered into the façade model as parameters (Figure 6.8). After these parameters are entered into the façade model, the size and position of all elements are immediately updated to reflect these classical proportions. When modelling classical historical buildings having these proportions already applied significantly reduces the amount of further editing to be carried out.

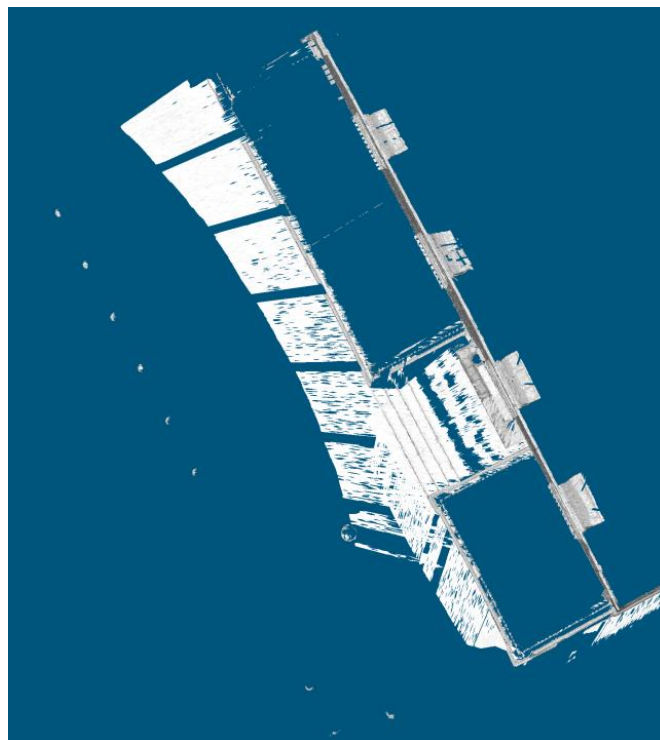


Figure 6.6: Orthographic image showing a plan view of a building façade and entrance created from laser scan survey data.

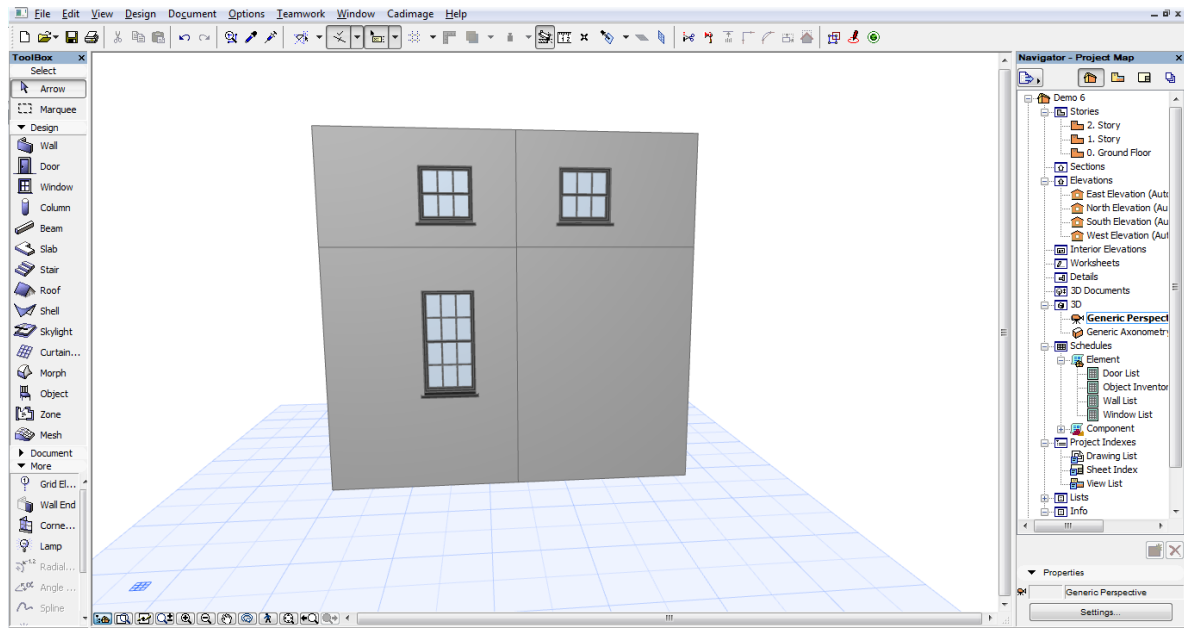


Figure 6.7: Initial configuration of a procedurally generated façade.

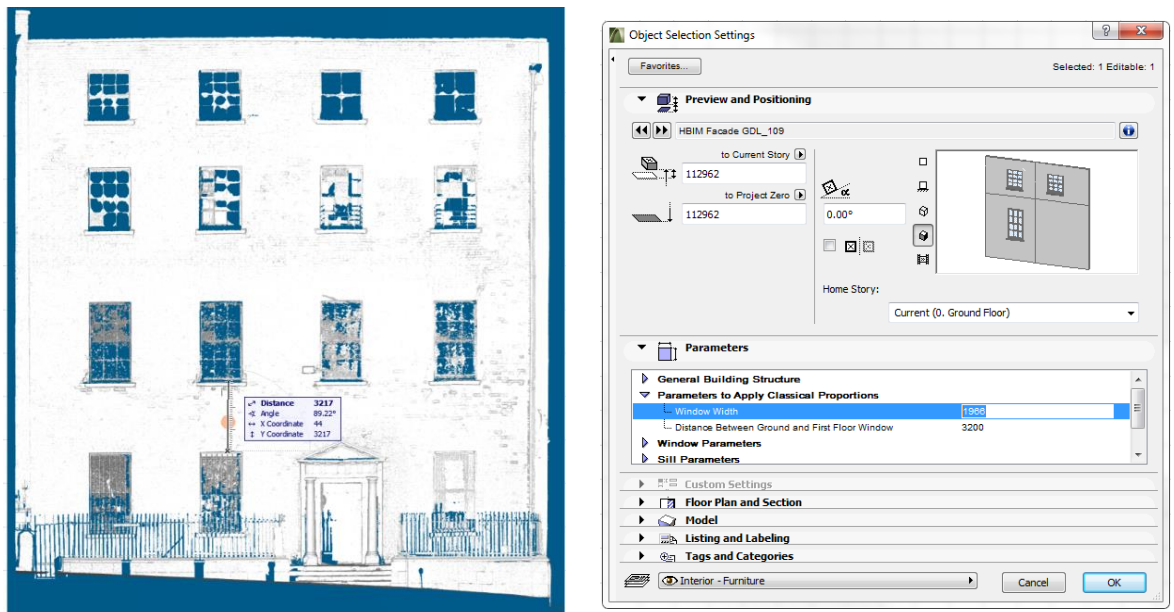


Figure 6.8: Measurements taken from survey data (left) and entered as parameters into dialogue box for procedural façade model.

The next stage involves specifying the façade structure. This is carried out by altering parameters for the structure of the façade either in a dialogue box or graphically in the 2D or 3D window by selecting and moving hotspots on the model. A user must specify



the number of storeys for the façade, the number of horizontal tiles and the position of the door on the ground floor. By altering these parameters the façade structure is procedurally generated (Figure 6.9 & Figure 6.10). Once the structure of the façade has been generated the façade model can then be overlaid with survey data and the initial estimates for objects provided by applying classical proportions can be assessed.

The final stage involves graphically editing objects on the procedural façade to accurately position elements relative to survey data. This is carried out in 2D or 3D while overlaying the model with survey data such as orthographic imagery as seen in Figure 6.11. In order to enable efficient editing, editable hotspot points can move multiple objects at once. This simultaneous editing allows users to quickly alter the heights of all windows on a floor simultaneously, the width of all windows above each other in a column simultaneously and also the position of all windows in a column or floor simultaneously. The distance between floors and columns can also be modified graphically to move multiple objects at once. Along with this editing with groups of objects, individual editing of objects is also possible to achieve high levels of accuracy relative to the survey data. Individual window corners can be modified by simply selecting a hotspot at the window corner and moving it to the desired new position (Figure 6.12).

As an element on a façade is modified all linked elements are also automatically updated as seen in Figure 6.13 where graphically editing a window opening automatically updates the combined ashlar block wall detail which calculates the required subtractions or additions using Boolean operations.



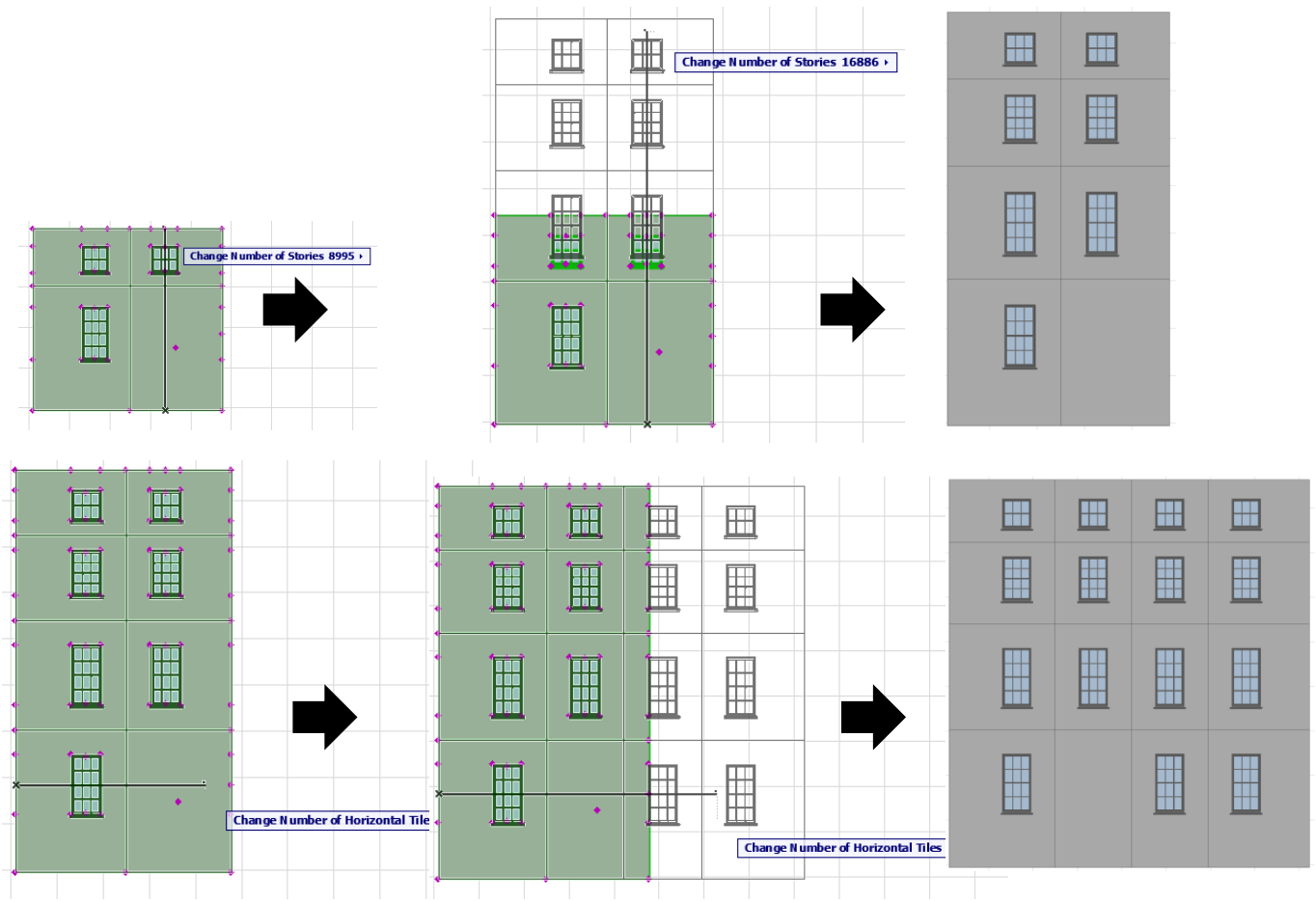


Figure 6.9: Graphical parameter editing to procedurally generate the structure of a façade. Parameter for the number of storeys edited (top) and the number of horizontal tiles edited (bottom).

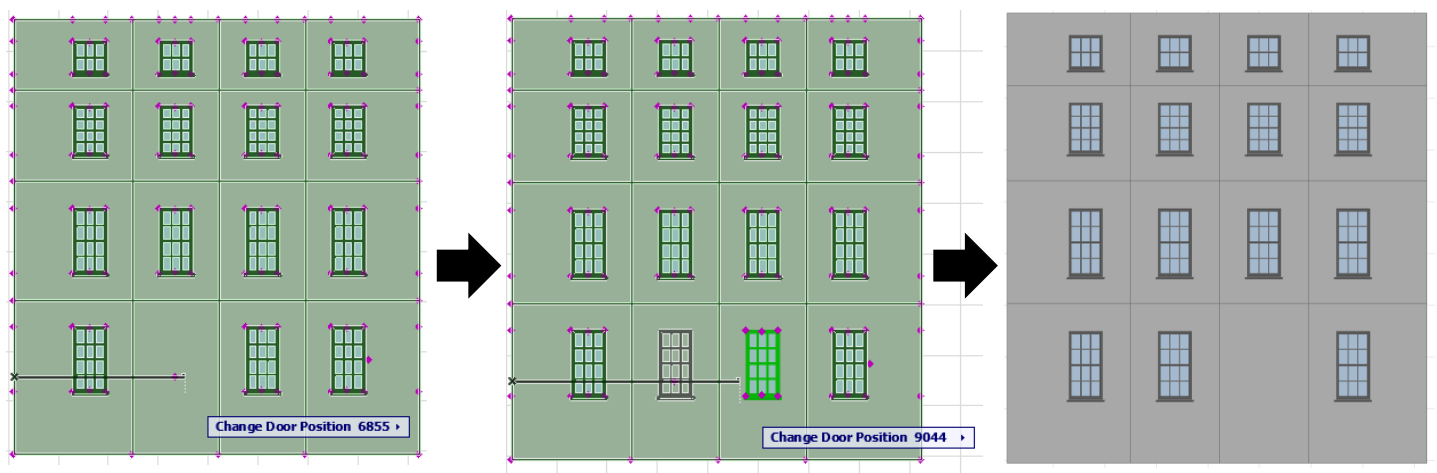


Figure 6.10: Graphical parameter editing to modify door position by moving a hotspot in the 2D window.



Figure 6.11: Orthographic image overlaid with procedural façade model for graphical editing of façade elements.

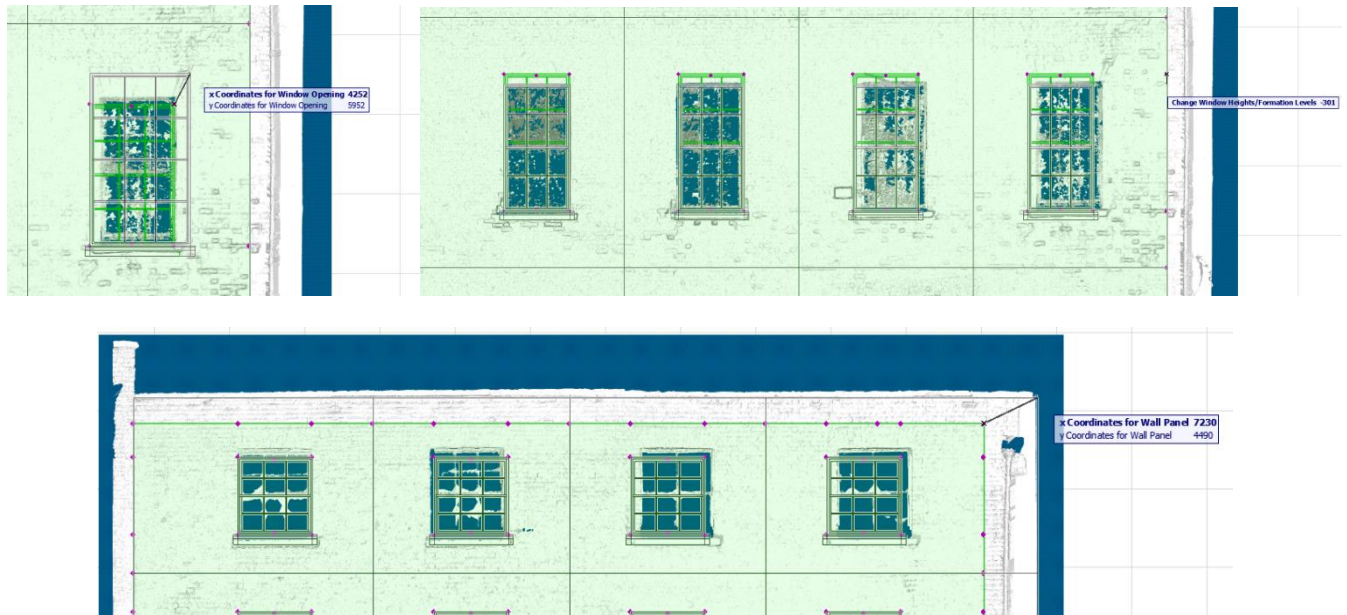


Figure 6.12: Graphical editing of objects on the procedural façade model. Editing of individual window (top left), editing of all window heights on a floor simultaneously (top right) and editing façade corner (bottom).

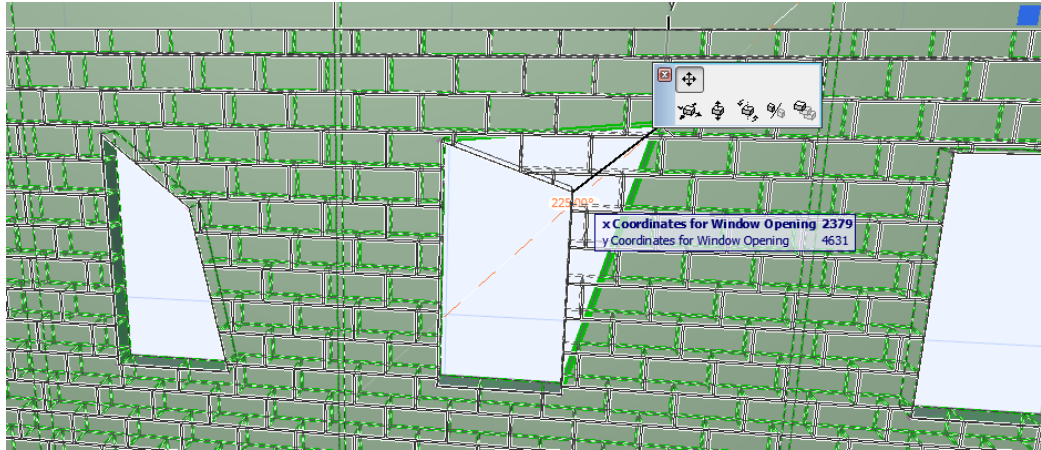


Figure 6.13: Graphically editing window opening which automatically updates linked elements such as ashlar block wall detail.

#### 6.4 Mapping to Survey Data – Prototype II

Similar techniques are adopted for mapping the second procedural building prototype to survey data. Procedural building models are first created by selecting one or multiple building footprints and then selecting the “Generate Building from Footprint” command from the new HBIM menu in the menu bar. Alternatively, users can select a “Draw Footprint and Generate Building” command from the same HBIM menu. This second command allows a user to draw a new polygon with user input that will be used to automatically generate the building model. These commands will automatically generate the walls for the building by applying the first three rules described in Section 4.3.

Next, a user can apply classical proportions to the building model by entering two measurements similar to the previous procedural façade prototype. Applying these classical proportions will update the heights of all floors and the position and size of all openings on each building side. Next users can adjust parameters to alter the building structure as required. This involves altering the number of storeys (Figure 6.14) and also

applying splits to automatically create the required number of openings on each building side. Users can then add objects to the building model by altering parameters from the dialogue box.

The final stage involves overlaying the generated building model with survey data such as orthographic imagery, segmented point clouds or cut sections. Users can then refine openings, floor heights or other objects by graphically editing hotspots in 3D or 2D windows (Figure 6.15). Hotspots can be used to edit groups of objects at once or individual objects on their own. Objects on the building can be accurately refined and positioned to specific survey data very quickly using this method. Hotspots are also included at the base of the building to change the building footprint (Figure 6.14). When a building footprint is changed all linked objects are instantly updated as seen in Figure 6.14.

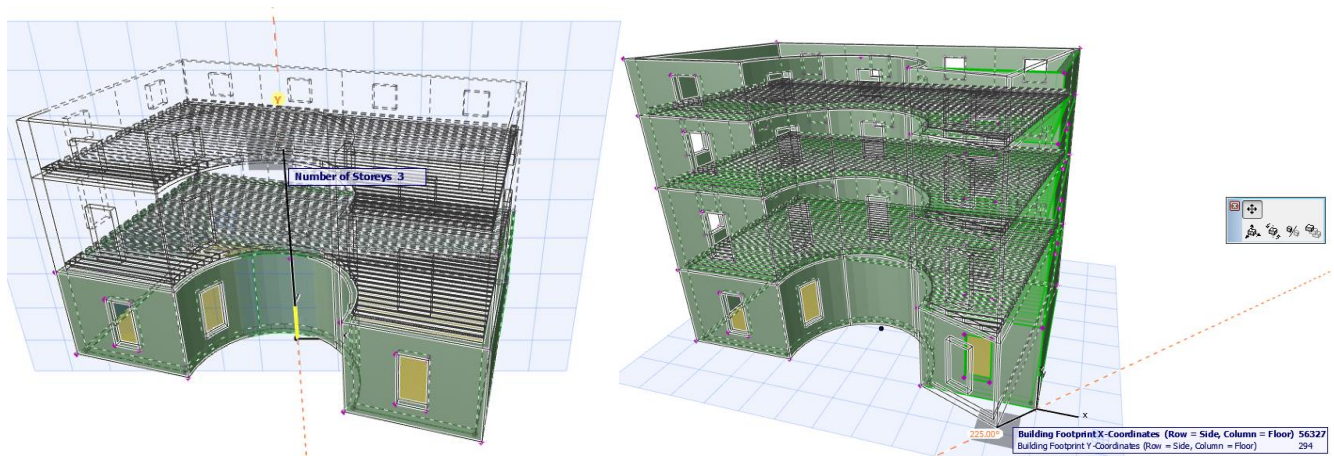


Figure 6.14: Graphical editing the “number of storeys” (left) and the building footprint (right) by clicking and dragging hotspots.

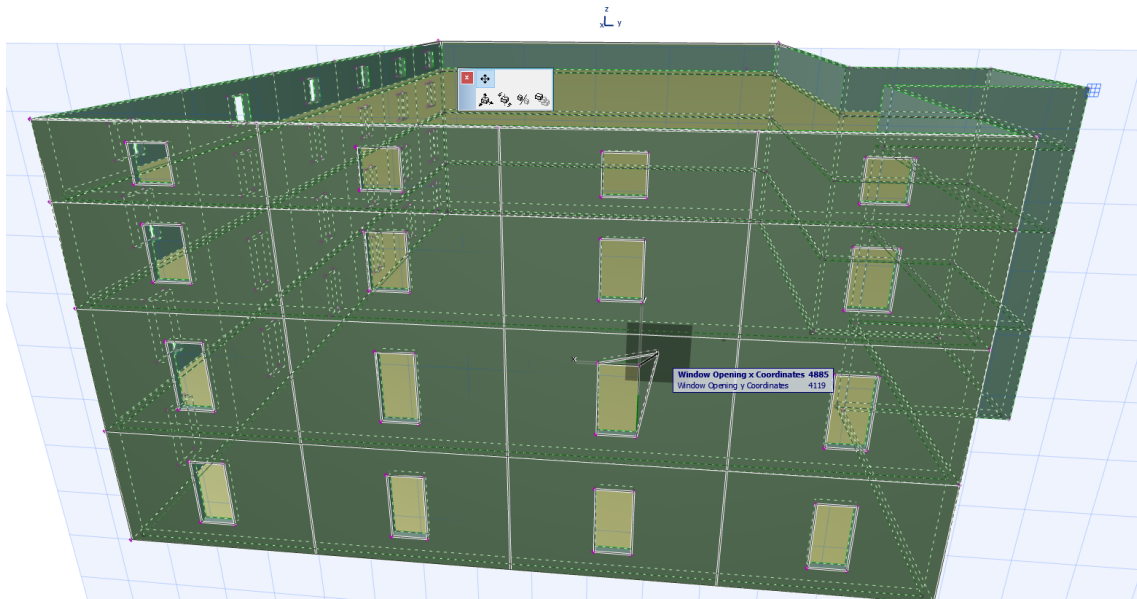


Figure 6.15: Graphically editing an individual window opening in the 3D window by clicking and dragging a window hotspot to a new position.

## 6.5 Mapping to Survey Data – Prototype III

For the third prototype, a non-vertical circular wall is first automatically generated from two or more cut-sections through a point cloud. This generates a true wall surface using polygon sections from a point cloud. Next users can apply classical proportions by again entering two measurements into a dialogue box similar to the first procedural façade prototype. Applying these classical proportions will set the heights of all floors and the size of all openings and objects that will be generated. Next users can adjust parameters to alter the building structure as required. This involves altering the number of storeys and also applying a split rule to divide a complete model or floor into any number of tiles. Users can then add objects to the procedural model by selecting objects in a dialogue box and choosing which tiles to place them on. Objects can be generated on all tiles in a model, all tiles on a floor or specific tiles. Specific arrangements of objects can also be generated such as alternating object on a floor.

Once the required geometry is generated on a wall structure then the position and sizes of objects can be refined to more accurately match survey data. Cut-sections, segmented point clouds or complete point clouds can be used to refine procedurally generated objects on a circular wall surface (Figure 6.16). Orthographic images which were used for previous prototypes are not suitable for the third prototype as the generated geometry is not planar.

First, the positions of objects are refined in a plan view (Figure 6.17). Refining objects on an irregular circular wall surface is based on editing angles instead of distances. New object positions are calculated based on angles from an average arc origin for the circular wall surface. To move a single object requires changing the angles from the average arc origin to both start and end angles of the object. Changing the width of an object involves changing either the start or end angle of an object from the average arc origin (Figure 5.22). Using these start and end angles to each object, simultaneous editing can also be carried out by changing all objects start and end angles at once which will move all objects simultaneously around a wall structure (Figure 6.18 & Figure 6.19). Once the start or end angles of an object are changed the new positions and widths of objects are calculated using the procedural rules described in Chapter 4.

When refining objects in a 2D plan view, all objects can first be edited simultaneously to roughly align all objects to survey data (Figure 6.17 & Figure 6.18). Next individual objects can be moved (Figure 6.20) or the width of a single object can be altered (Figure 6.21). After the positions of objects are refined, next the formation levels (Figure 6.22) and heights (Figure 6.23) can be edited with hotspots from 2D section windows or a 3D window. Graphical hotspots enable the heights and formation levels of all objects on a floor to be edited simultaneously or individually as required. Using graphical hotspots



to interactively refine objects simultaneously or individually enables very efficient editing and mapping of generated geometry to specific survey data.

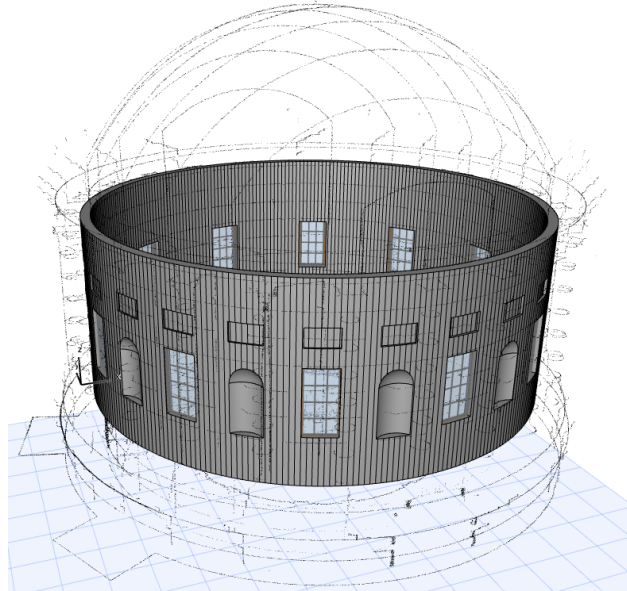


Figure 6.16: Cut-sections from a point cloud used to refine procedurally generated geometry (3D window).

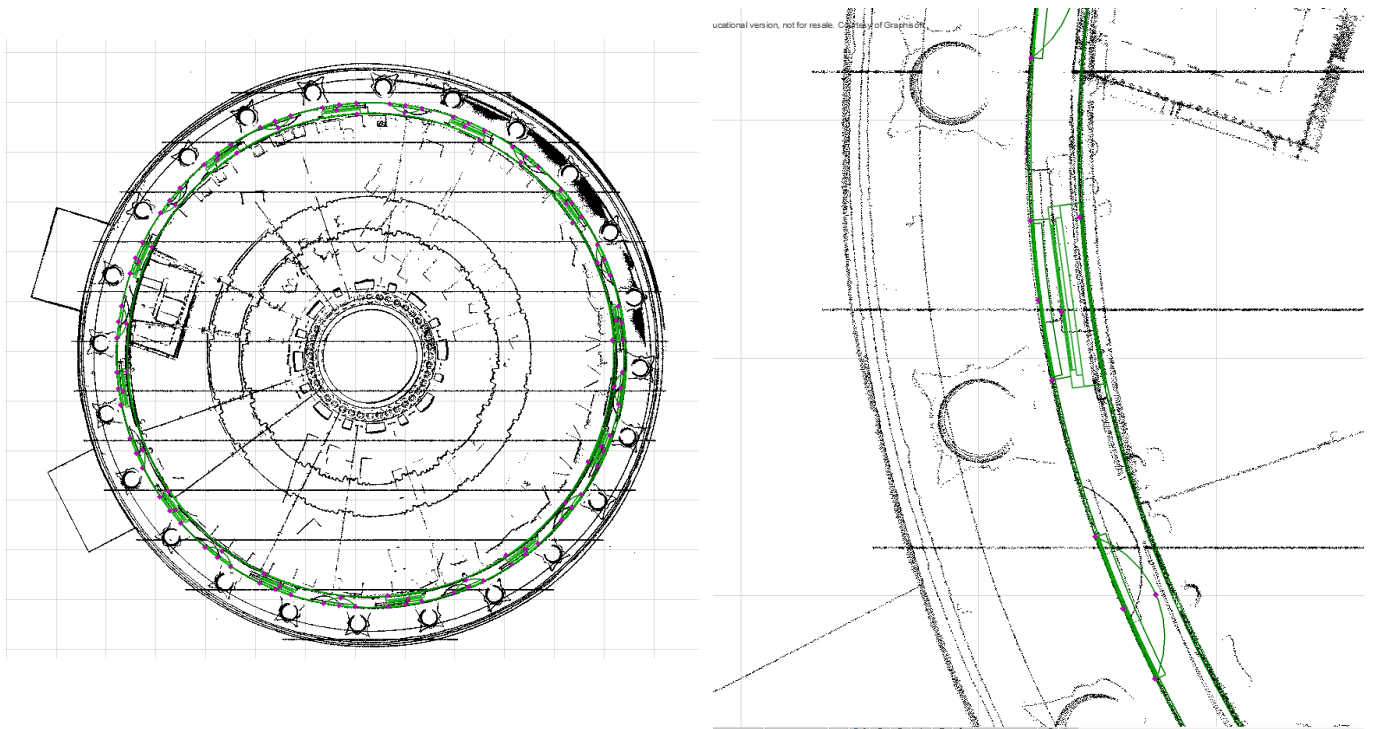


Figure 6.17: Cut-sections from a point cloud used for refining procedurally generated objects in a 2D (plan) window.

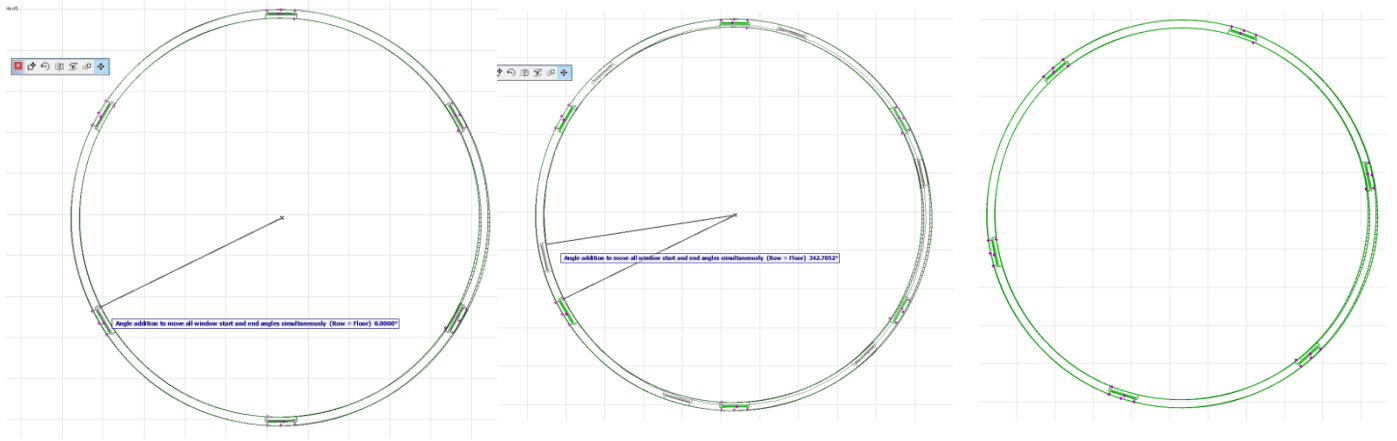


Figure 6.18: Simultaneous editing of all procedurally generated objects in a model based on angles from an average arc origin. An angle based editable hotspot is used to rotate all objects around the irregular wall structure (2D plan window).

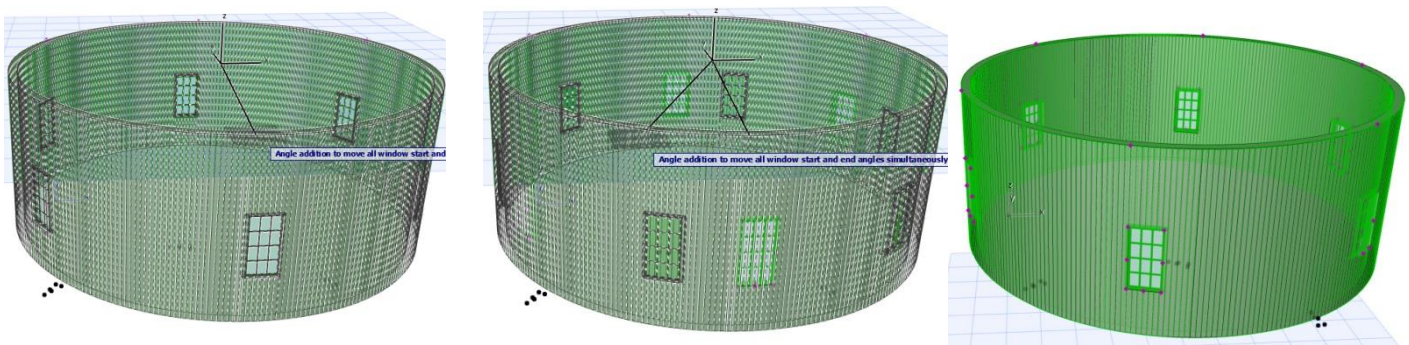


Figure 6.19: Simultaneous editing of all procedurally generated objects in a model based on angles from an average arc origin. An angle based editable hotspot is used to rotate all objects around the irregular wall structure (3D window).

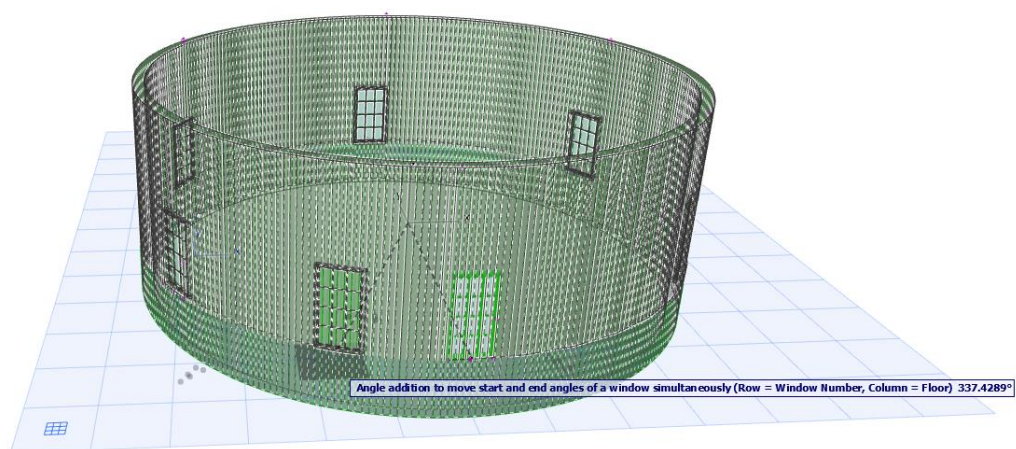


Figure 6.20: A hotspot located in the middle of an object used to move a single object at once. This alters an individual objects start and end angles to move the object around a circular wall structure.



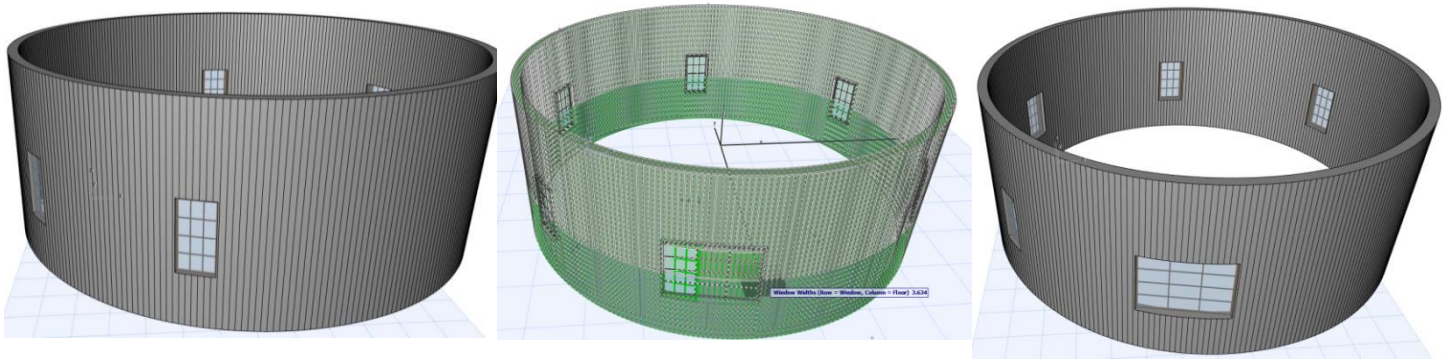


Figure 6.21: An angle type hotspot at either side of an object is used edit the width of an individual object. The width is calculated based on a new start or end angle of the object.

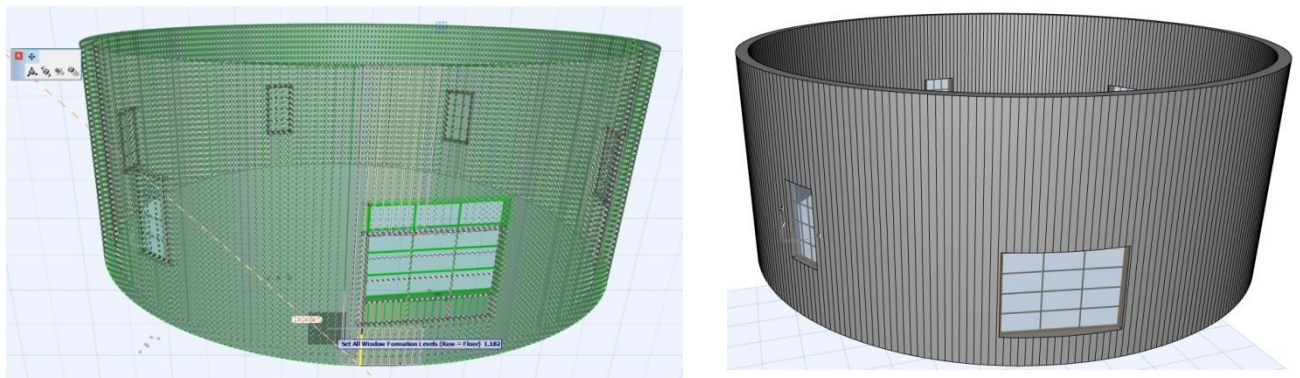


Figure 6.22: A hotspot located at the bottom of an object is used to change the formation level of an object.

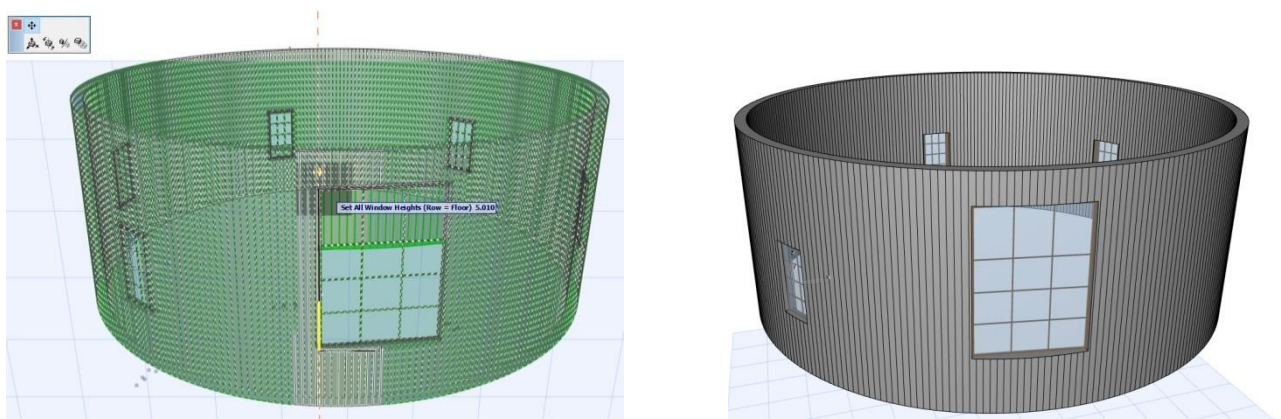


Figure 6.23: A hotspot located at the top of an object is used to change the height of an individual object.

## 6.6 Conclusion

This chapter outlined the process of mapping the procedural modelling prototypes to specific survey data for modelling existing buildings. To overcome limitations of BIM software, new tools have been developed to enable survey data to be accurately imported in its true position using a common coordinate reference system. Once survey data is imported and accurately positioned in BIM software then the procedural modelling prototypes can be used for fast, efficient and accurate modelling from survey data. The process involved in each prototype is a semi-automatic process where geometry for a required building arrangement is first automatically generated and then manually refined to specific survey data. Prototypes II and III also enable further automation by using 2D building footprints and cut-sections to generate wall surfaces.

When using procedural modelling to generate objects in a model, architectural rules are used to estimate the position and size of objects. This greatly reduces the amount of further editing required for classical architecture. When manually refining objects on a procedural model, editable hotspots are used to interactively edit parameters of objects directly from the model by clicking and dragging hotspots. Objects can be edited in groups for efficient and fast editing. Objects can also be edited individually for high accuracy results.

# **Part III**

## **EVALUATION AND TESTING**

**Part III Summary:**

Part III of this thesis contains two chapters which describe the methodology for testing and validating the new concepts of procedural HBIM. In the first chapter of Part III, two case studies are undertaken to fully implement and validate the procedural HBIM concepts using real world applications. The second chapter of Part III describes three tests undertaken for further validation of the procedural HBIM concepts and prototypes.

## **Chapter Seven: Case Studies**

### **7.1 Introduction**

This chapter describes two case studies which were undertaken to test the newly developed procedural modelling prototypes. This validation with real world applications is the final implementation of the developed prototypes. The first case study undertaken involved documenting the buildings on Henrietta Street, one of the earliest Georgian streets in Dublin. The second case study undertaken was a restoration project of Ireland's main courts building, the Four Courts, which is also located in Dublin. For this second case study, the third procedural modelling prototype was used to produce fast and accurate documentation that would form a basis for a proposed conservation intervention.

### **7.2 Case Study 1 - Henrietta Street Dublin**

#### **7.2.1 Introduction and Background to Case Study**

Henrietta Street is an 18th century Georgian Street located in Dublin, Ireland. Henrietta Street is one of the earliest Georgian streets in Dublin and is of great historical significance. The street was developed by Luke Gardiner and constructed between 1730 and 1820. The buildings on Henrietta Street are great examples of classical style architecture and were originally seen as city palaces. During the 19th and 20th century the street fell into disrepair and despite recent restoration work, there are still a number of buildings that need urgent attention.

#### **7.2.2 Data Collection and Pre-Processing**

The entire street was recorded using laser scanning and image acquisition methods. Eight scans were carried out with 10mm resolution using a Trimble GS200 terrestrial laser scanner as seen in Figure 7.1. Common targets were surveyed at a higher

resolution (2mm) in common areas between scans for registration of scans. Five to eight common targets were surveyed between scans to ensure accurate registration results during processing.

A number of pre-processing steps were carried out on the scan data before modelling with BIM software. These pre-processing steps included registration, segmentation, filtering, triangulation, texturing, orthographic image creation and the generation of cut-sections. All processing of scan data was carried out using Trimble Realworks software. Figure 7.2 shows the final point cloud recorded for Henrietta Street.



Figure 7.1: Trimble GS200 laser scan used to record data for generating an as-is BIM of Henrietta Street.

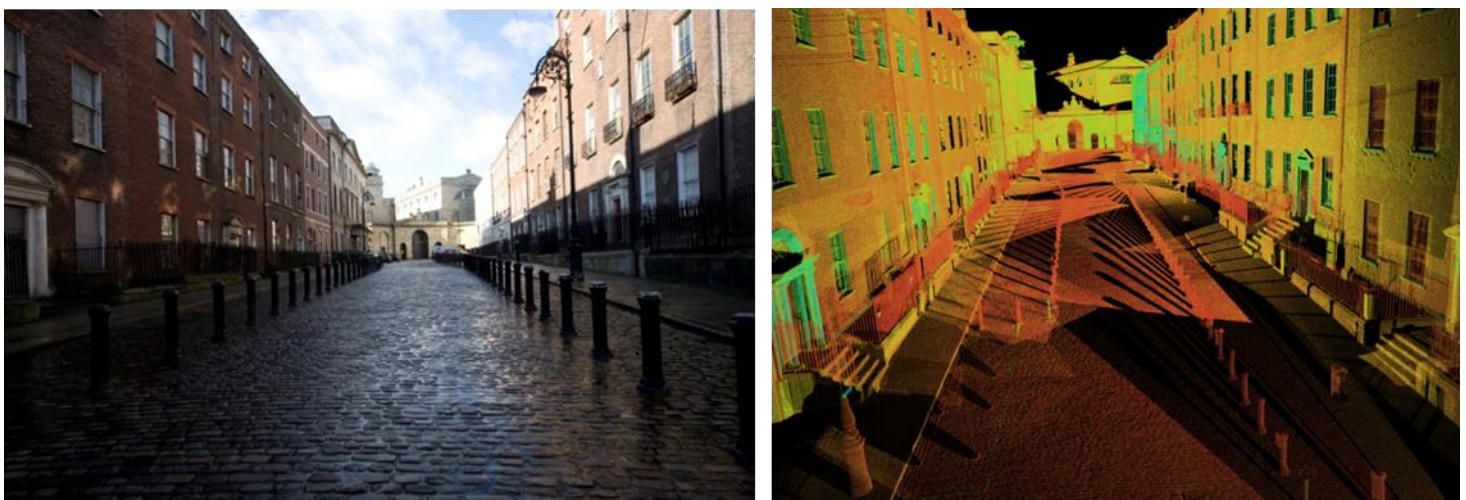


Figure 7.2: Henrietta Street, Dublin (left) and point cloud of street coloured by intensity (right).

### **7.2.3 Generation of BIM Geometry with the Procedural Façade Prototype**

Next data from the survey including segmented point clouds, orthographic images and cut-sections were imported into the ArchiCAD BIM software for modelling. The newly developed tools for importing geo-referenced orthographic imagery and cut-sections were used to accurately position all survey data within the BIM software (Figure 6.1). The procedural façade prototype along with additional HBIM parametric library objects (Murphy et al., 2013), were then used to generate accurate building information models from the survey data. Manual modelling of each individual façade on this street would be a very time-consuming task and would involve measuring, positioning and adjusting hundreds of objects to reconstruct the street. Instead, each façade structure was automatically generated with the procedural façade prototype. Parameters were adjusted for each façade to automatically generate the correct building arrangement with the correct number of storeys, number of openings, door position and types of objects. Once each façade was automatically generated, the position of façade elements were quickly refined using efficient simultaneous editing or individual editing as required. Additional HBIM parametric library objects (Murphy et al., 2013) were also added to complete models as required (Figure 7.3). These HBIM parametric library objects were manually mapped to survey data and façade models. Figure 7.3 shows the building information model generated with the procedural façade prototype and HBIM library objects. A video of the captured point cloud and reconstructed HBIM can also be seen from the link below.

**<https://youtu.be/81EJCnxUcQo>**

### **7.2.4 Documentation and Further Applications of Data**

A model generated with the procedural façade prototype and HBIM parametric library objects can be used for more than just visualisation. Along with the geometric representation, the generated HBIM also contains further information regarding



relationships, semantics and attributes of building components. This additional information facilitates further analysis and enables the model to be used as a management tool for building operations and maintenance. Numeric data, lists of objects, components and their attributes along with bills of quantities can be automatically generated from a HBIM to facilitate economic and valuation analysis. Figure 7.4 shows an example of lists of objects, components and a bill of quantities automatically generated from the HBIM for Henrietta Street. This data includes information about objects such as area, volume, length, thickness, quantities and position within a model.



Figure 7.3: Building information model created using the procedural façade prototype and HBIM library objects for Henrietta Street, Dublin.

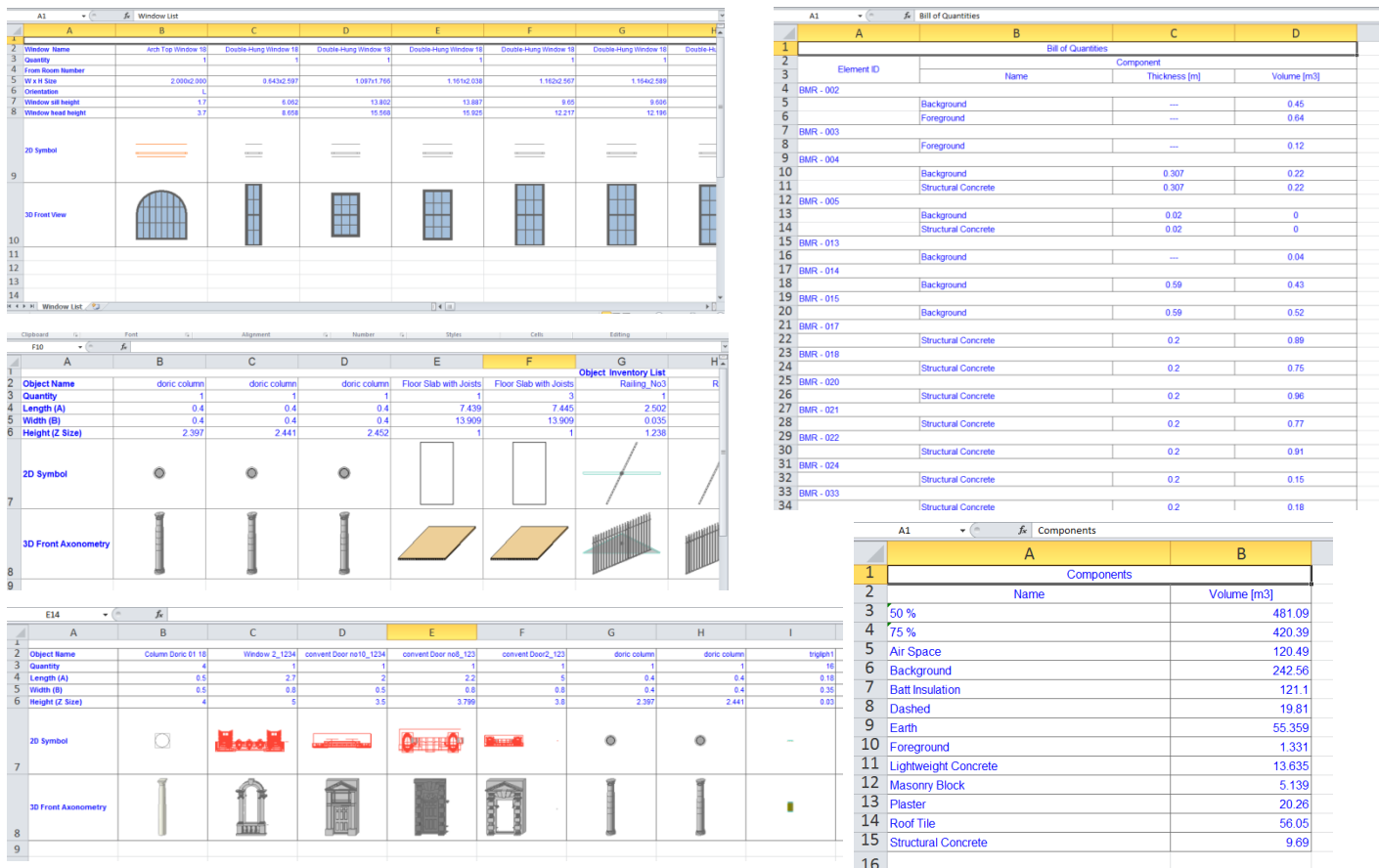


Figure 7.4: Lists of objects, components and bill of quantities generated from the HBIM of Henrietta Street.

Where conservation or restoration work is to be carried out on an object or structure, conventional orthographic or 3D survey engineering drawings are required. Once a HBIM has been generated and precisely mapped to survey data, 2D and 3D documentation can be automatically generated from the 3D model. This includes plans, elevation drawings, sections and 3D perspectives as shown in Figure 7.5 and Figure 7.6.

A model created with the procedural HBIM prototypes and library objects can also be used for energy analysis. Using the EcoDesigner STAR extension for ArchiCAD BIM software it is possible to perform energy analysis with HBIM data. A HBIM must first be converted to a Building Energy Model (BEM) by enhancing the HBIM with information relating to energy evaluation. A workflow from a HBIM to BEM requires three steps. Firstly spaces within a HBIM need to be organised into thermal blocks. This

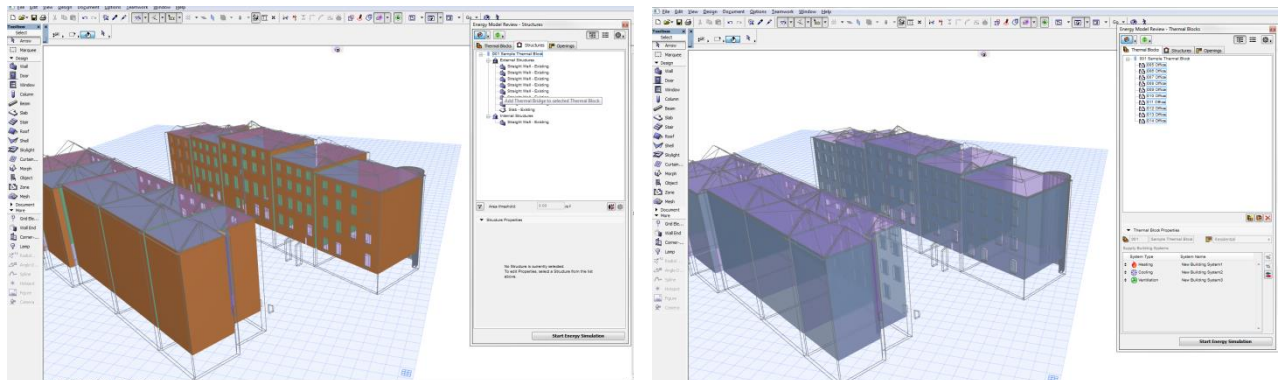


The image displays a set of architectural drawings for a building. On the left, there is a section drawing showing the internal structure, including a staircase and a small arched entrance at the base. To its right is a floor plan showing the layout of the ground floor, with a central staircase and several rooms. Below the section drawing is a small elevation of a building facade, showing a central entrance with a pediment and a small balcony. To the right of this is a vertical section of a wall, showing the internal structure and a window. On the far right is a large elevation drawing of a building facade, showing a symmetrical design with a central entrance, a pediment, and a series of windows arranged in a grid. The drawing is labeled 'Elevation' and 'Section'.

179



Figure 7.6: Automated 2D documentation from BIM model.



## Energy Performance Evaluation

### 3-15 Henrietta Street HBIM

#### Key Values

##### General Project Data

Project Name: Henrietta Street HBIM  
City Location: Dublin  
Latitude: 53° 21' 10" N  
Longitude: 6° 16' 13" W  
Altitude: 0.00 m  
Climate Data Source: IRL\_Dublin...90\_IWEC.epw  
Evaluation Date: 29/09/2014 15:18:02

##### Building Geometry Data

Gross Floor Area: 1975.03 m<sup>2</sup>  
Treated Floor Area: 1764.88 m<sup>2</sup>  
External Envelope Area: 5301.47 m<sup>2</sup>  
Ventilated Volume: 28941.16 m<sup>3</sup>  
Glazing Ratio: 5 %

##### Building Shell Performance Data

Infiltration at 50Pa: 1.09 ACH  
Outer Heat Capacity: 491.67 J/m<sup>2</sup>K

##### Heat Transfer Coefficients

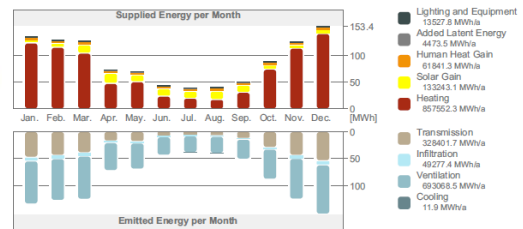
Building Shell Average: 1.15 [W/m<sup>2</sup>K]  
Floors: ---  
External: 0.97 - 3.37  
Underground: ---  
Openings: 3.41 - 4.04

##### Specific Annual Values

Net Heating Energy: 552.90 kWh/m<sup>2</sup>a  
Net Cooling Energy: 0.00 kWh/m<sup>2</sup>a  
Total Net Energy: 552.91 kWh/m<sup>2</sup>a  
Energy Consumption: 560.57 kWh/m<sup>2</sup>a  
Fuel Consumption: 560.57 kWh/m<sup>2</sup>a  
Primary Energy: 23.00 kWh/m<sup>2</sup>a  
Fuel Cost: --- GBP/m<sup>2</sup>a  
CO<sub>2</sub> Emission: 1.66 kg/m<sup>2</sup>a

Degree Days  
Heating (HDD): 3690.05  
Cooling (CDD): 614.67

#### Project Energy Balance



#### Thermal Blocks

Thermal Block	Zones Assigned	Operation Profile	Gross Floor Area m <sup>2</sup>	Volume m <sup>3</sup>
001 Sample Thermal Block	10	Residential	1975.03	28941.16
Total:	10		1975.03	28941.16

#### Environmental Impact

Source Type	Source Name	Primary Energy kWh/a	CO <sub>2</sub> emission kg/a
Renewable	External Air	16	0
Secondary	Electricity	40596	2922
Total:		40613	2922

Figure 7.7: Energy analysis performed on HBIM data for Henrietta Street using EcoDesigner STAR extension for ArchiCAD BIM software.

### **7.2.5 Review of Case Study**

This case study of Henrietta Street showed the capabilities and flexibility of the first procedural façade prototype. The prototype was capable of generating the required building arrangements for all the fourteen buildings on the street. Graphical editing allowed for all generated objects to be quickly and accurately refined using correctly positioned orthographic images. Applying classical proportions also resulted in faster editing as the amount of editing required was reduced. Using this procedural façade prototype it was also possible to represent non-uniform objects such as window openings. Using existing software tools and library objects within ArchiCAD, it was not possible to accurately model all window objects. This is due to deformation as some openings on the street do not have perfect 90-degree rectangular openings. Some sash windows also had an irregular number of panes that could not be represented with existing ArchiCAD library objects. The new prototype enabled these irregular openings to be accurately modelled and the window objects generated by the prototype could also accommodate for irregular numbers or sash panes. An inclination angle could also be set for facades that were not perfectly vertical. Overall the procedural façade prototype enabled more efficient and accurate modelling of the building façades on Henrietta Street when compared to existing manual methods within the ArchiCAD software.

A limitation of the current procedural façade prototype for this case study was that not all required objects on a façade could be generated by the prototype. Additional parametric HBIM objects needed to be manually added to complete façade models. Other sides of buildings, roofs and street furniture were also manually created. A solution to this would be to extend the façade prototype with more parametric objects to enable a more efficient modelling process. Subsequent procedural modelling prototypes provide a greater number of objects such as niches, columns and floor slabs. The case study also showed the capabilities for different types of analysis after a building

information model is created with this prototype. This includes automatic lists of objects, components and bill of quantities for economic analysis. Plans, sections, elevations and 3D views can be automatically generated for documentation and conservation analysis. Energy analysis can also be performed to estimate net heating energy, net cooling energy, energy consumption, fuel consumption, CO2 emissions, emitted energy per month and environmental impact.

### **7.3 Case Study 2 - The Four Courts Dublin**

#### **7.3.1 Introduction and Background to Case Study**

A restoration project of Ireland's main courts building, The Four Courts, was used as a second case study to test the new procedural HBIM developments. The Four Courts is a late 18<sup>th</sup>-century classical building located in Dublin, Ireland. The Four Courts was designed by architect James Gandon and constructed between 1786 and 1802. The building was partially destroyed by fire in 1922 during the Civil War in Ireland which occurred during the early establishment of the Republic (Figure 7.8). During this time the original dome which was supported by timber collapsed (Figure 7.9).

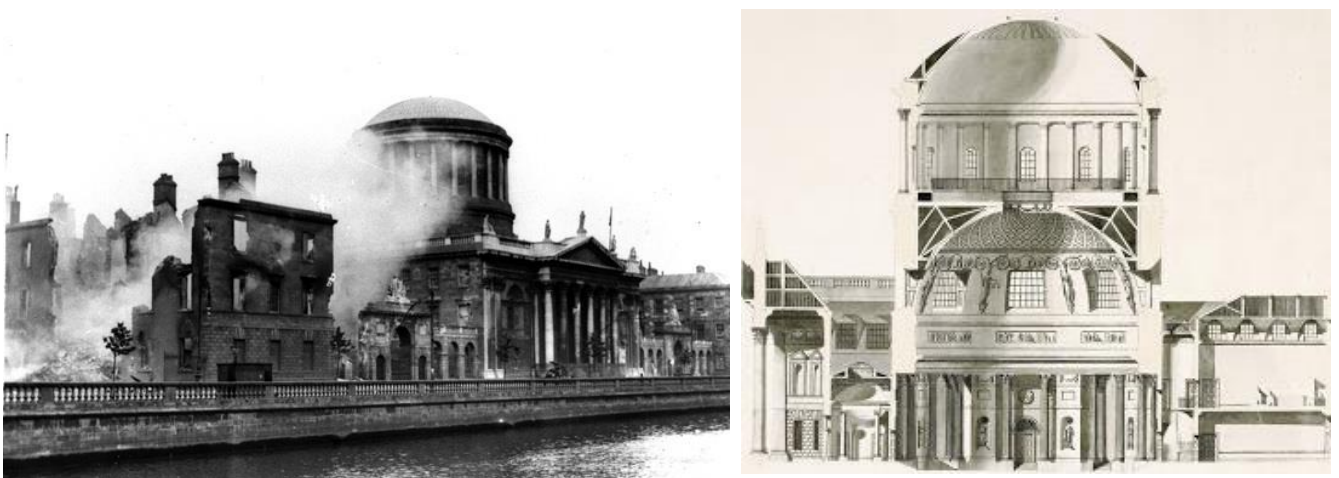


Figure 7.8: War damage to the Four Courts (left) (Cashman, 1922a) and 18<sup>th</sup>-century section drawing of the Four Courts by architect James Gandon (right) (Gandon, 1700s).



Figure 7.9: Damage to the Four Courts after the bombardment by the Free State Army, during the Civil War on 30 June 1922 (Cashman, 1922b).

### 7.3.2 Previous Restoration Works

The dome was rebuilt in the late 1920s. This new dome was constructed using reinforced concrete which was poured over a period of thirty-six hours. Despite being damaged in the Civil War, the stone capitals from the original building were saved and reincorporated in the new dome. Twenty-four columns support the dome, all of which are slightly different due to weathering and fire damage. When the dome was being rebuilt, the original capitals were rotated so that the exposed or damaged sides faced inwards. This was possible as the capitals had been carved on all sides. Subsequent renovations were also carried out in the 1940s when structural problems resurfaced. A ring of steel inserted in the 1920s when the dome was rebuilt had rusted and concrete was used to cover it up.

The effects of the Civil War damage to the building are once again a threat to the structural stability of parts of the building. In 2011, a large section of one of the capitals that top each of the Corinthian columns supporting the dome broke away and fell onto a roof below (Byrne, 2015). Investigations showed that this collapse was caused by further rusting of the steel ring encircling the concrete dome (Figure 7.10). This resulted



in the weight of the dome bearing down on the capitals below causing the delicate, carved Portland stone to crack (Figure 7.10).



Figure 7.10: Part of a damaged Corinthian capital that broke away and fell onto the roof below (top left). Damaged Corinthian capital (top right) and rusted steel ring encircling the dome (bottom).

### 7.3.3 Current Restoration Work

The current restoration work required dealing with all structural issues before moving on to finer issues such as the capitals themselves. Conservation and restoration work to the capitals would require both mortar repairs and new stone carving. The concrete on the inside of the dome also required resurfacing. The first stage of the work involved erecting scaffolding to assess and measure the extent of the damage (Figure 7.11). A laser scan survey was commissioned to provide accurate measurements for documentation. In order to produce engineering drawings and to carry out further analysis, it was decided that parts of the laser scan survey would be converted into a

Building Information Model. The next sections describe the laser scan survey and the use of the procedural HBIM prototypes to generate an accurate building information model. The aim of this part of the project was to use HBIM to illustrate virtually the current extent of the damage/decay. This information would then be used as a basis for the proposed conservation interventions.

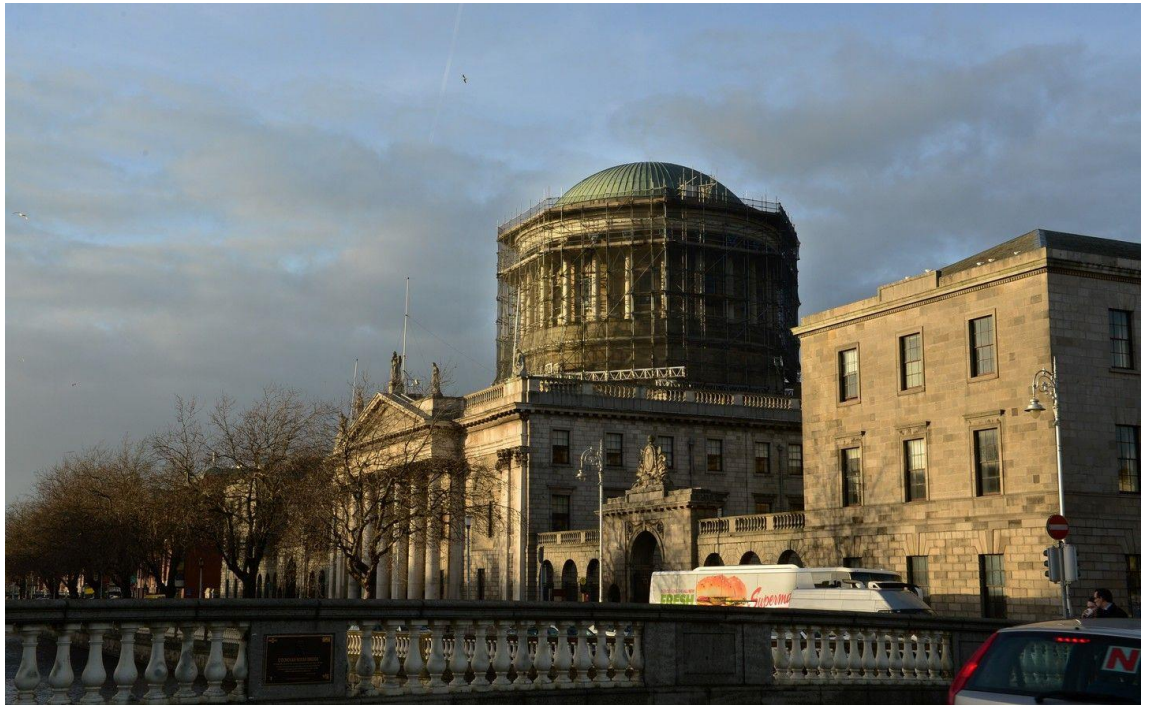


Figure 7.11: Exterior view of the Four Courts Dublin where the dome is undergoing major restoration (Byrne, 2015).

#### **7.3.4 Laser Scan Survey & Pre-Processing**

A complete laser scan was carried out on the internal and external structure using a Leica HDS C10 scanner. Separate laser scan surveys were carried out, before and after scaffolding was erected around the dome. This allowed the drum and columns to be surveyed before the scaffolding was put in place along with detailed scans of the capitals and upper dome after scaffolding was in place. Both laser scan surveys carried out were registered together in pre-processing using common targets that were set-up on site. Along with registering separate scans, other pre-processing steps involved filtering

and segmenting point clouds and generating orthographic images and cut-sections. All pre-processing of scan data was carried out using Leica Cyclone software. Results of this laser scan survey are shown in Figure 7.12 to Figure 7.14.

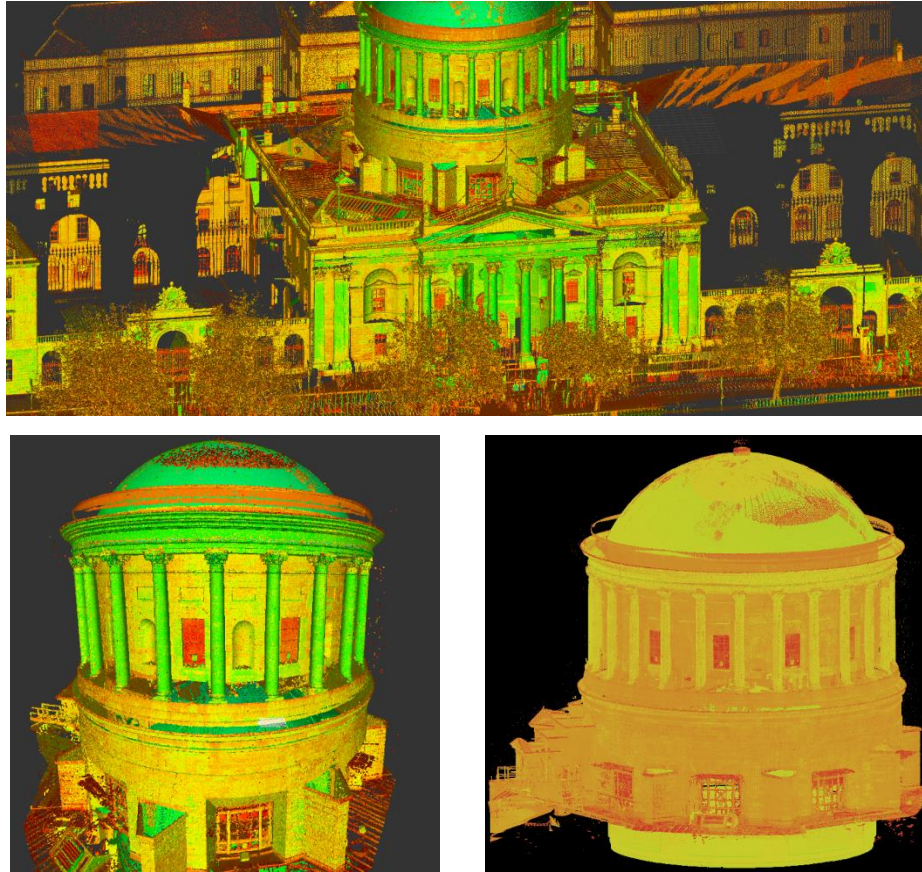


Figure 7.12: Point cloud of The Four Courts showing points coloured by their intensity values.



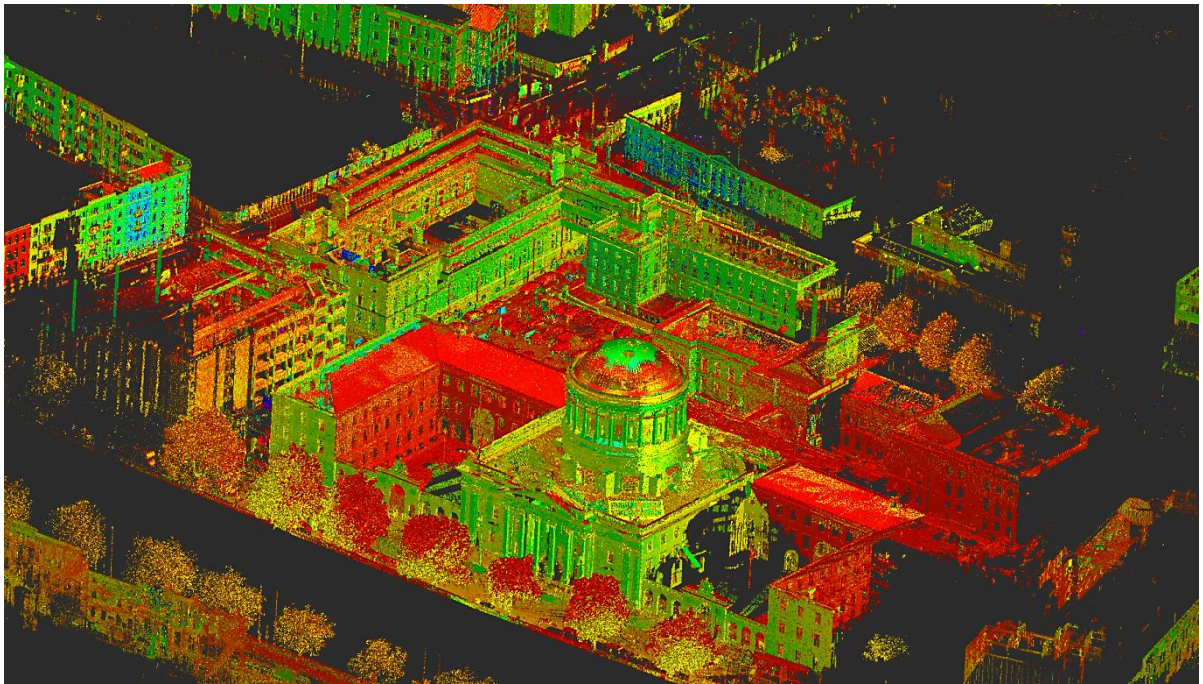


Figure 7.13: Complete registered point cloud of the Four Courts and surrounding buildings coloured by intensity values.

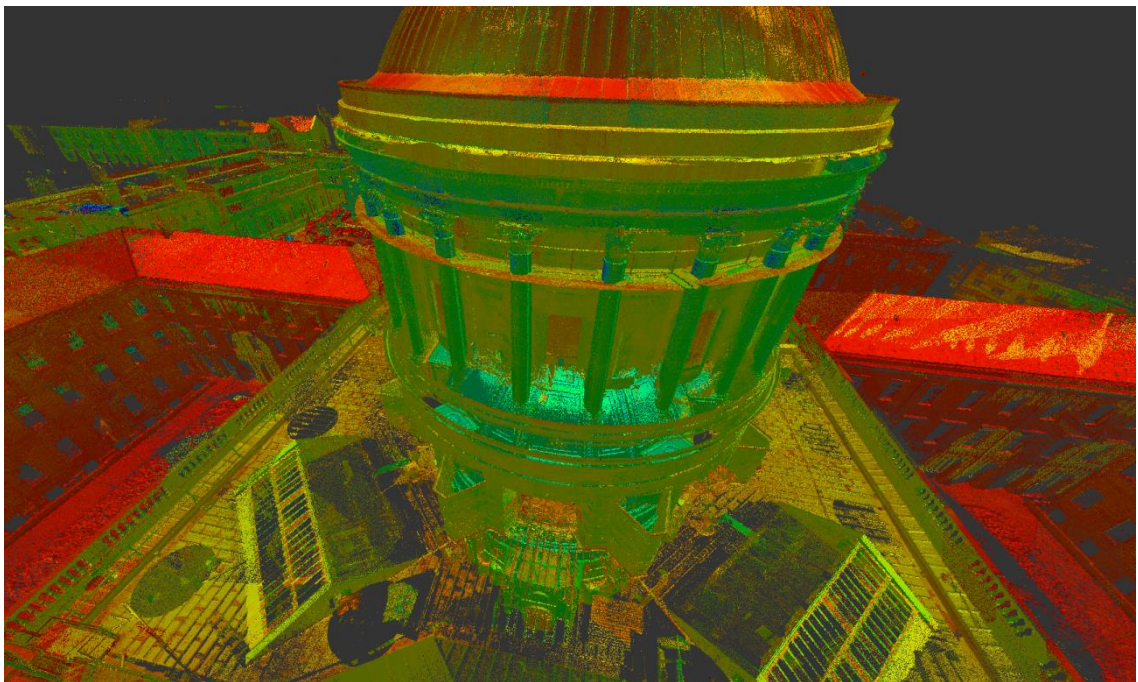


Figure 7.14: Point cloud showing detailed scans of capitals and the upper dome after scaffolding was put in place.

### 7.3.5 Generation of BIM Geometry with the Irregular Procedural Building Prototype

For this case study, the accuracy of the resulting HBIM was crucial in order to perform structural analysis on the damaged dome and drum. This required the irregular circular walls of the drum which supports the dome to be modelled representing its true condition in order to show up any areas of deformation or warping. This was possible with the third Irregular Procedural Building prototype.

A series of horizontal cut-sections at different heights were taken from the point cloud of the drum and used as input data for the HBIM procedural rules (Figure 7.15). The HBIM procedural rules for prototype III were then used to automatically generate the irregular BIM wall geometry which connects each horizontal cut-section. This allowed the non-vertical circular walls to be accurately and automatically modelled representing its true condition. It would not be possible to model this wall as accurately with existing ArchiCAD BIM tools as it is not possible to model non-vertical circular walls within this software. Cut-sections were also used to generate the internal wall surface with the first procedural rule to ensure that both internal and external wall surfaces were accurately represented (Figure 7.16).

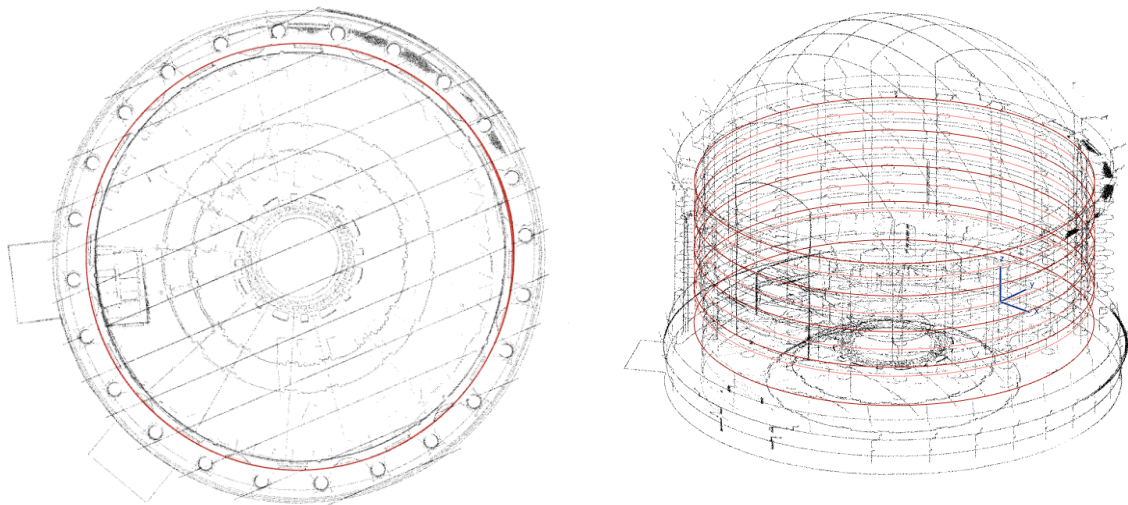


Figure 7.15: Cut-sections taken through the point cloud for the Four Courts dome and drum. Polygon sections shown are in red.

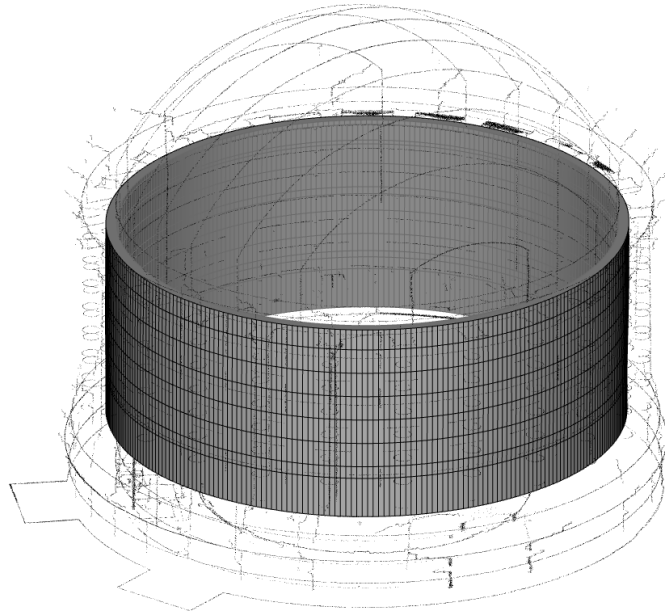


Figure 7.16: Irregular non-vertical circular walls automatically generated from polygon cut-sections.

Once the walls were generated further parameters were then adjusted to automatically generate the required arrangement of objects. Parameters were set to create one floor, split into twenty-four tiles with alternating arch-top niche and window objects. Rectangular niche objects were also automatically generated in each tile (Figure 7.17). The positions and size of these objects were then graphically refined in a group and also individually as required. Objects were first positioned in plan and then heights were matched to point cloud data in a 3D view.

Additional objects such as floors and the internal concrete dome surface were automatically generated as triangulated mesh objects from sampled and segmented point clouds (Figure 7.17). Other components including column shafts and beams being supported by columns were also automatically generated from cut-sections using the first rule, “procedural surface generation from cut-sections” (Figure 7.18 & Figure 7.19). Standard column library objects and beams could also have been used but this would not show the true condition of each object which contains deformation due to weathering and fire damage from the Civil War. The capabilities of the procedural rules



to model the drum, columns and beams from cut-sections showed the versatility of the new procedural rules. As column shafts and beams were automatically generated from cut-section, the resulting geometry is positioned in its true position and doesn't require further manual editing. Figure 7.20 shows the combined components of the Historic Building Information Model (HBIM) generated for the Four Courts comprising of the drum, internal dome surface, floor surface, column shafts and beams encircling the dome.

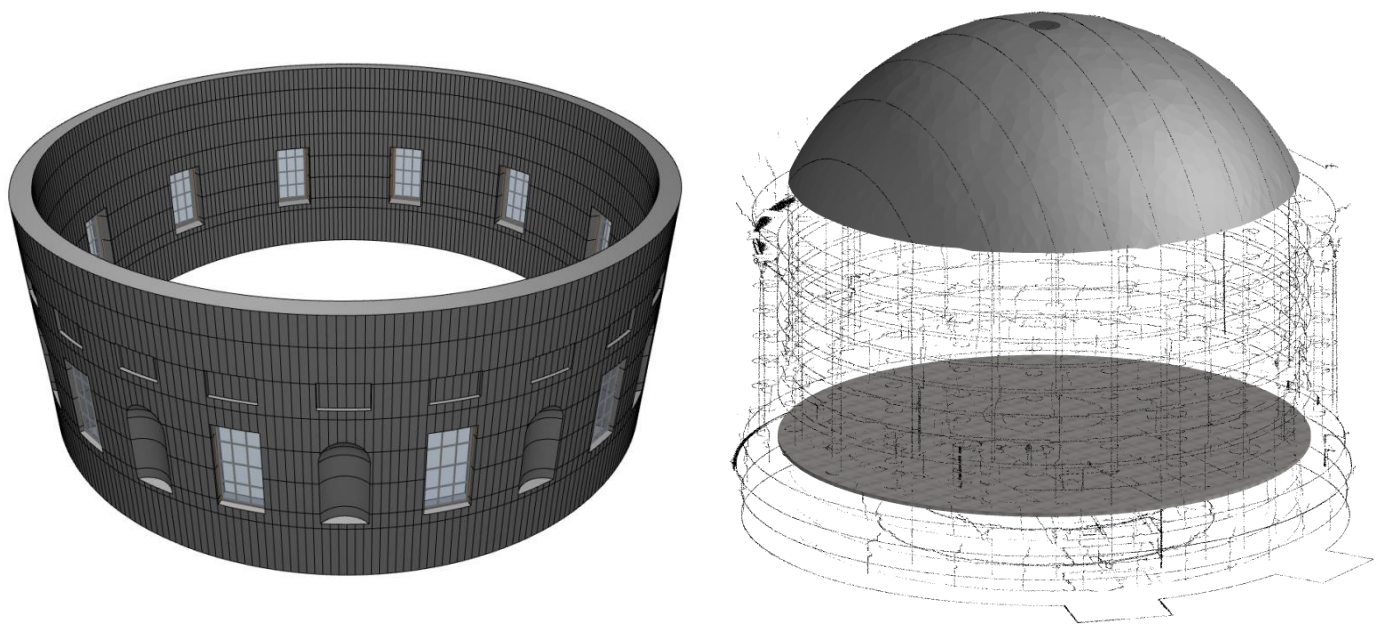


Figure 7.17: Non-vertical wall generated with 24 tiles containing alternating niche and window objects (left). Internal dome surface and floor surface automatically generated as mesh surfaces from segmented point clouds (right).

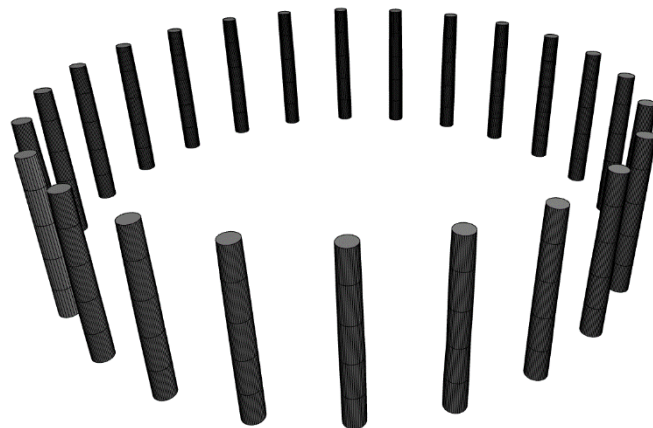


Figure 7.18: True condition of column shafts automatically generated from cut-sections.

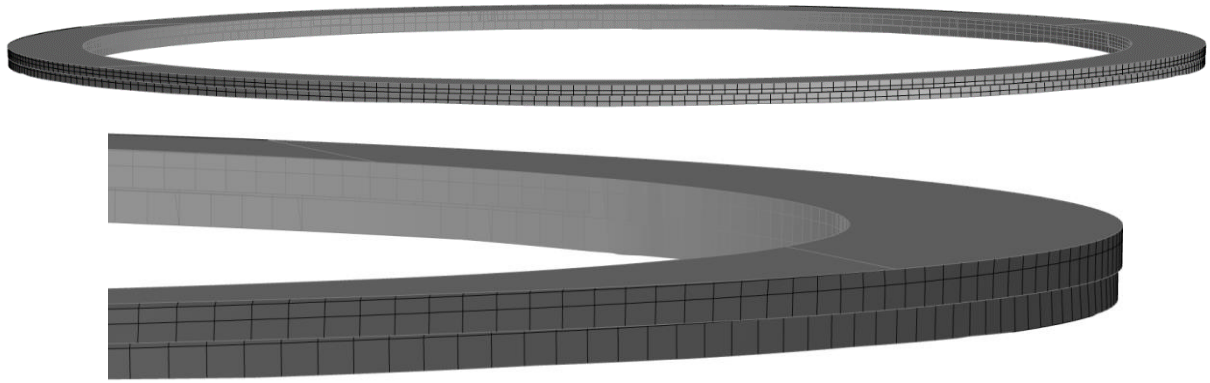


Figure 7.19: True condition of beam encircling dome automatically generated from cut-sections.

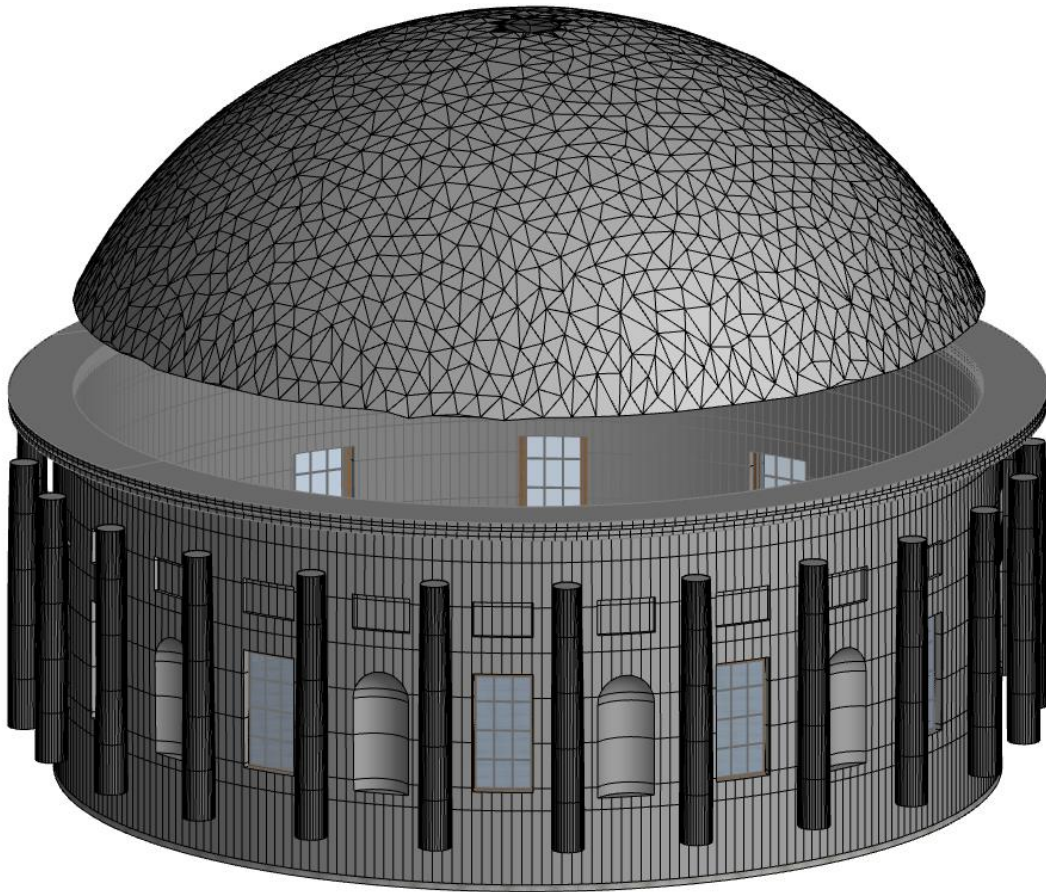


Figure 7.20: Historic Building Information Model (HBIM) components for the Four Courts automatically generated using the new HBIM procedural rules.

The Corinthian capitals can be represented in BIM software with standard library objects from the HBIM library of objects. With these library objects, however, it is

difficult to show deformation that is unique to a particular capital. An alternative solution was tested which involved importing triangulated mesh models for each Corinthian capital which is created directly from the point cloud or alternatively using photogrammetric techniques (Figure 7.21). These mesh models show the true condition of each capital but can contain a very large number of surfaces to represent the object. In order to enable the triangulated mesh models to be imported into ArchiCAD, each mesh model was resampled to reduce the number of surfaces to a more manageable size. Figure 7.22 shows an example of a mesh model representing the true condition of a capital combined with the previously generated HBIM for the Four Courts. The methodology for testing and validating the accuracy of different BIM components generated for this case study with the procedural rules is discussed in Chapter 8. The results of these accuracy tests are analysed in Chapter 9.



Figure 7.21: Triangulated mesh models of a Corinthian capital from the Four Courts which was captured using photogrammetry. A photo textured mesh is shown on the left and an un-textured mesh model is shown on the right.

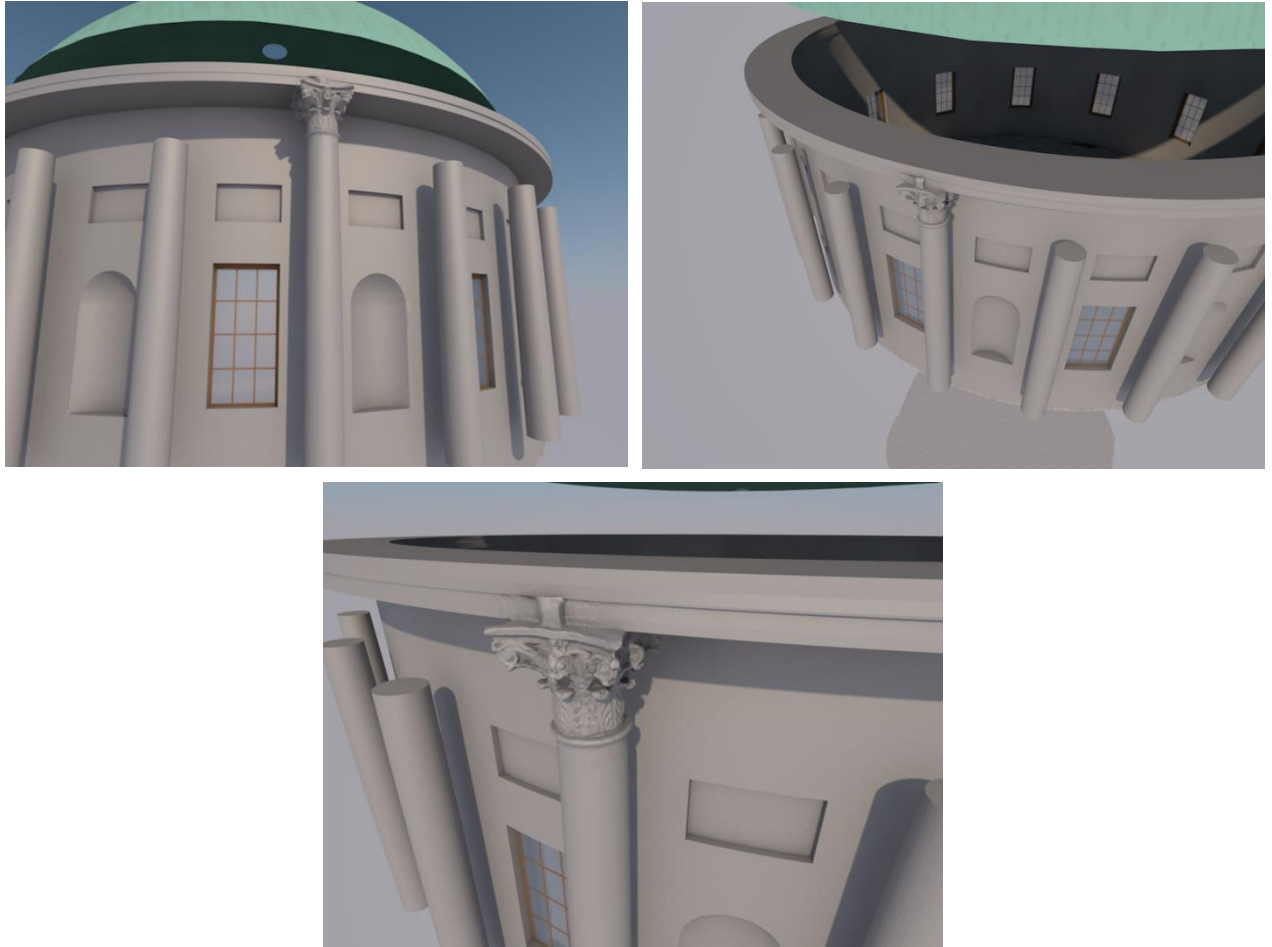


Figure 7.22: HBIM of the Four Courts with imported triangulated mesh model for a Corinthian capital representing its true condition.

### 7.3.6 Documentation Results & Analysis

The final HBIM for the Four Courts was used to produce various 2D and 3D documentation such as plans, section and elevations (Figure 7.23 to Figure 7.28). Figure 7.23 shows a plan vector drawing automatically generated from the 3D model. From this plan view it is possible to identify the verticality of walls, columns and beams and identify areas where walls, columns or beams are leaning (Figure 7.24). Figure 7.25 to Figure 7.27 shows vertical sections through the drum and dome. From vertical sections, it is possible to precisely measure the extent of a leaning wall at any point in the model. Figure 7.27 shows the variance between vertical wall lines (red dashed lines) and the true condition of the wall. A horizontal distance measured from the top of the actual

wall to the vertical wall line is 90mm at this particular location (Figure 7.27). Figure 7.28 shows a coloured elevation vector drawing also automatically generated from the 3D model.

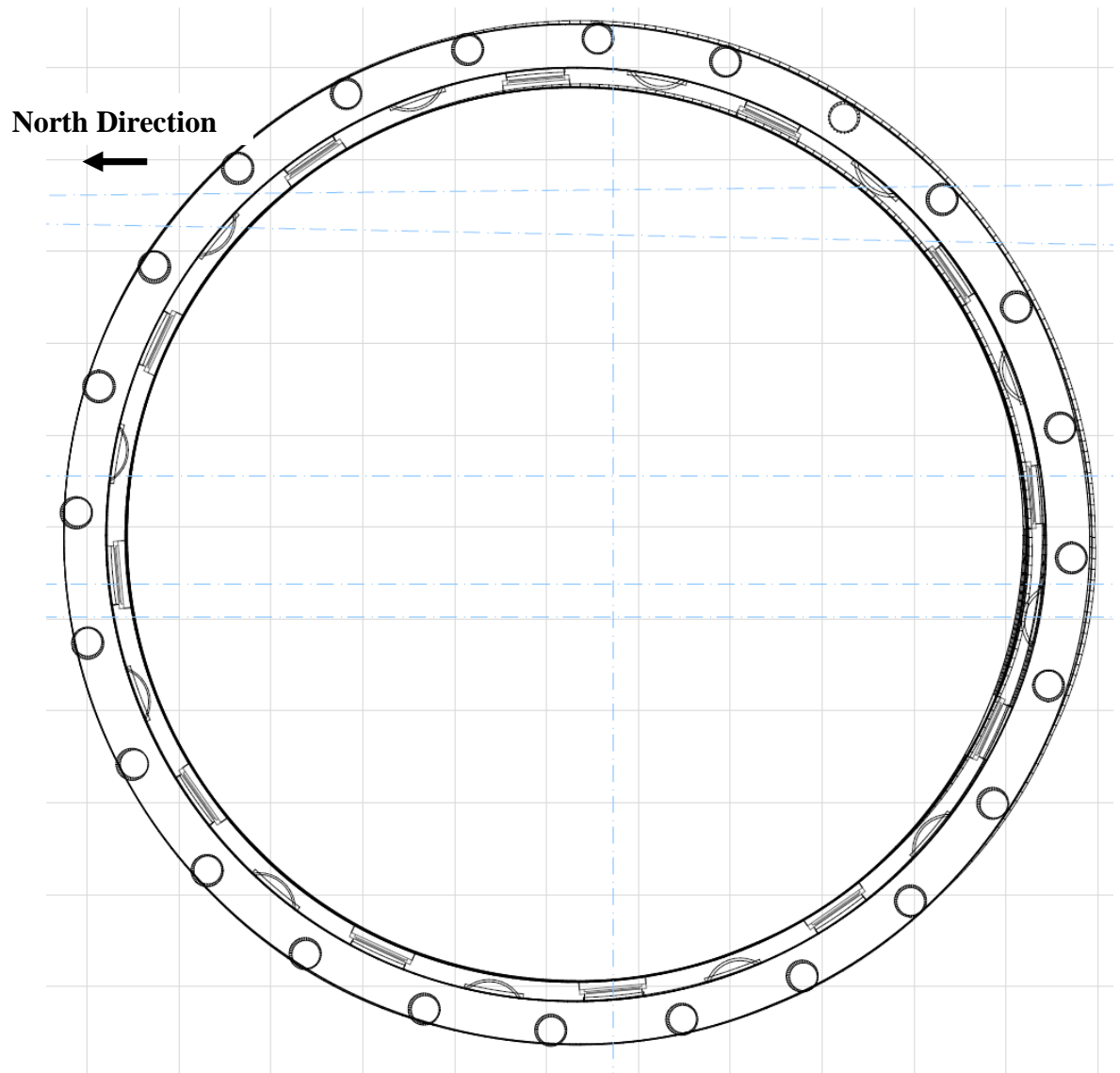


Figure 7.23: 2D plan vector drawing automatically generated from the 3D model for the Four Courts, Dublin.



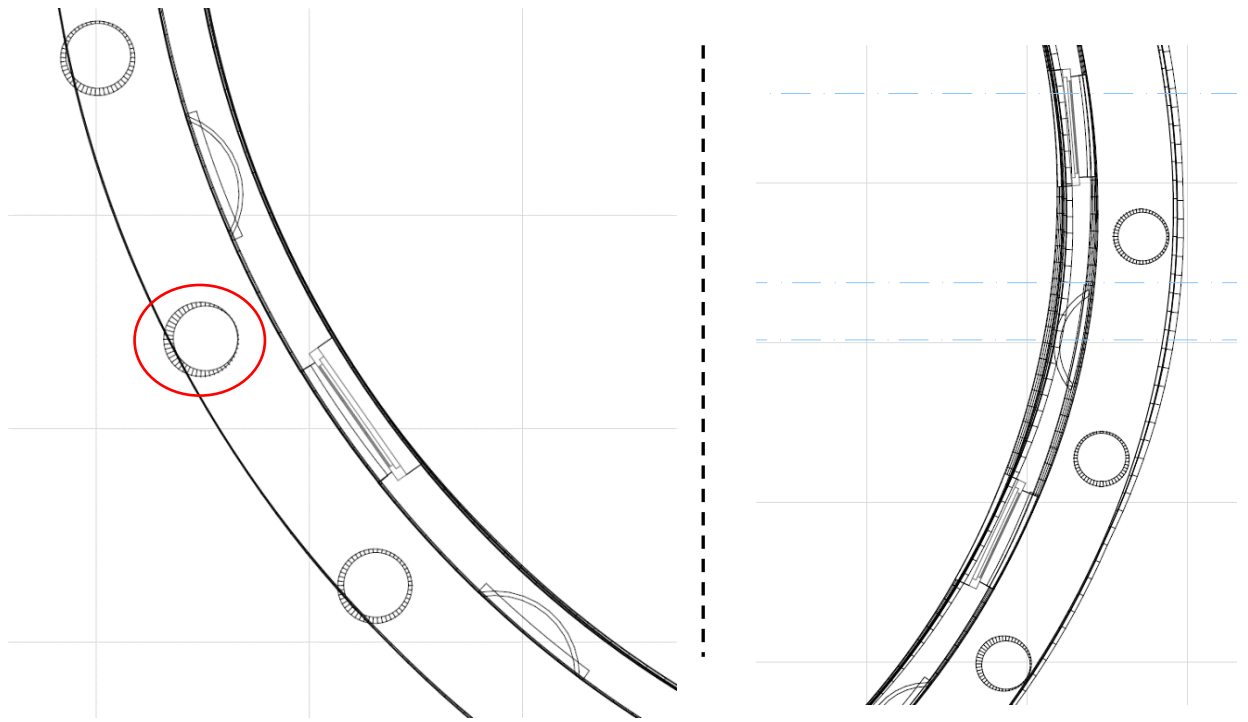


Figure 7.24: 2D plan vector drawing showing a column that is leaning (left). Documentation shows vertical walls (left) and areas of the wall and beam that are leaning (right).

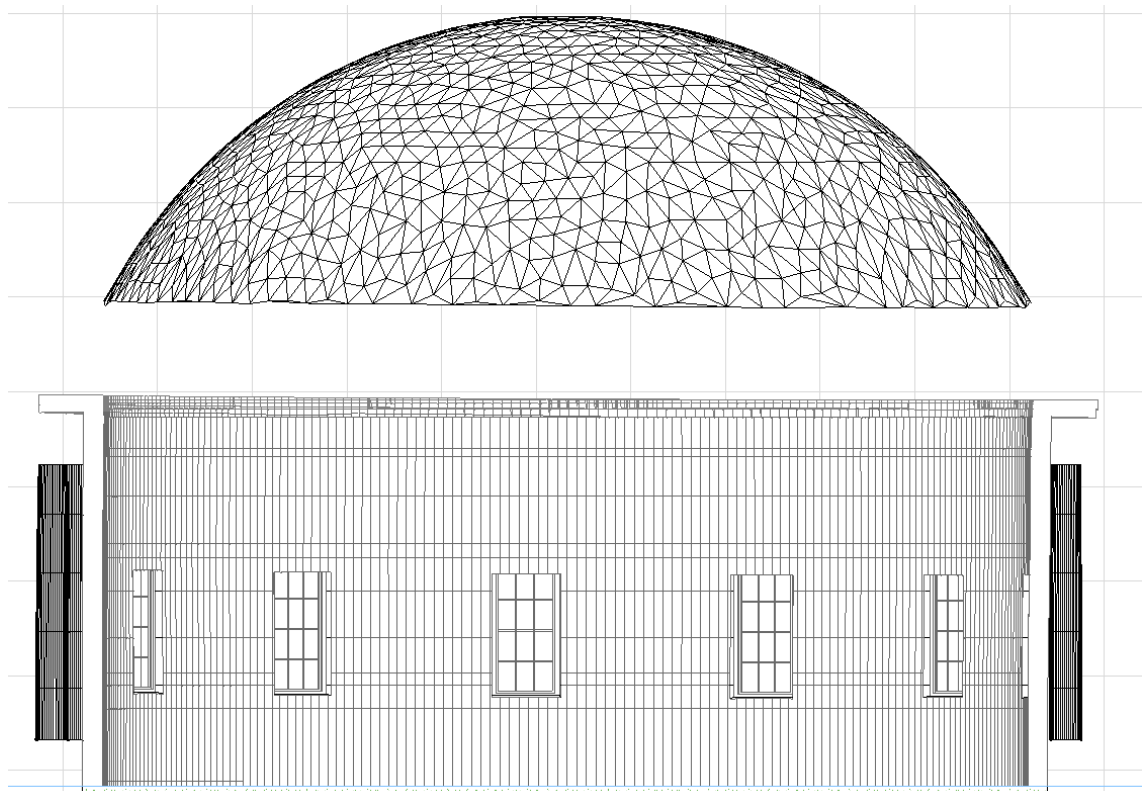


Figure 7.25: Section through the Four Courts drum and dome automatically generated from the 3D model.

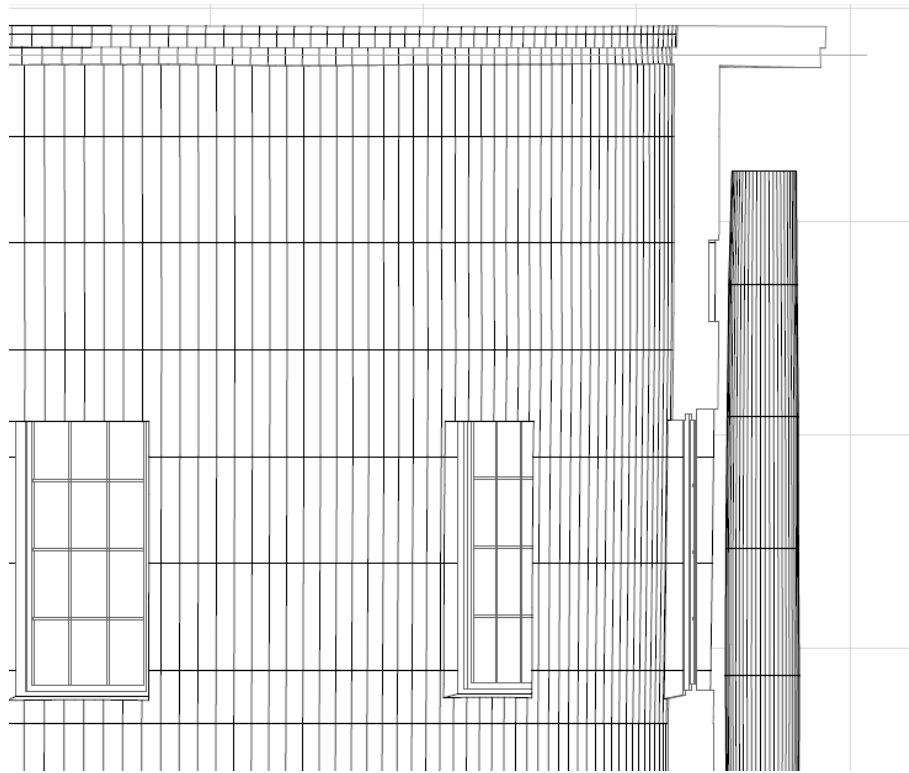


Figure 7.26: Section through the Four Courts drum automatically generated from the 3D model.

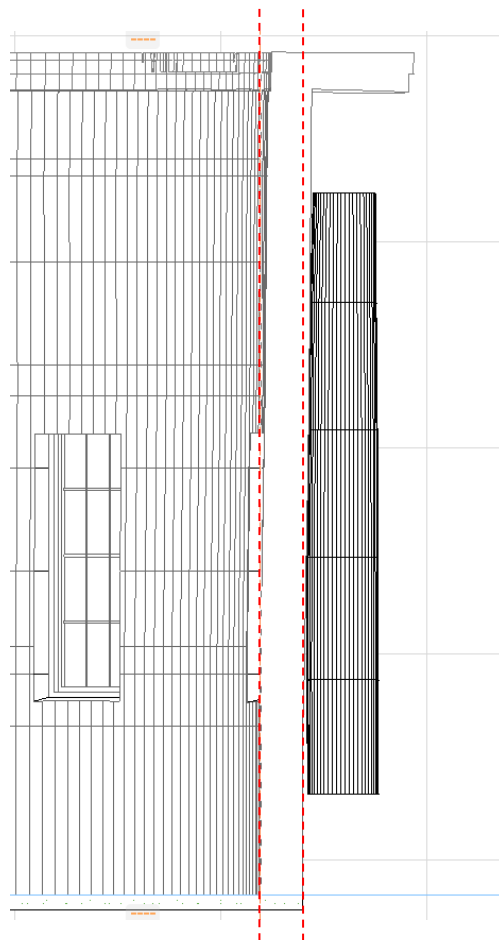


Figure 7.27: Section showing the true condition of walls which contains deformation (vertical wall lines shown by dashed red lines).

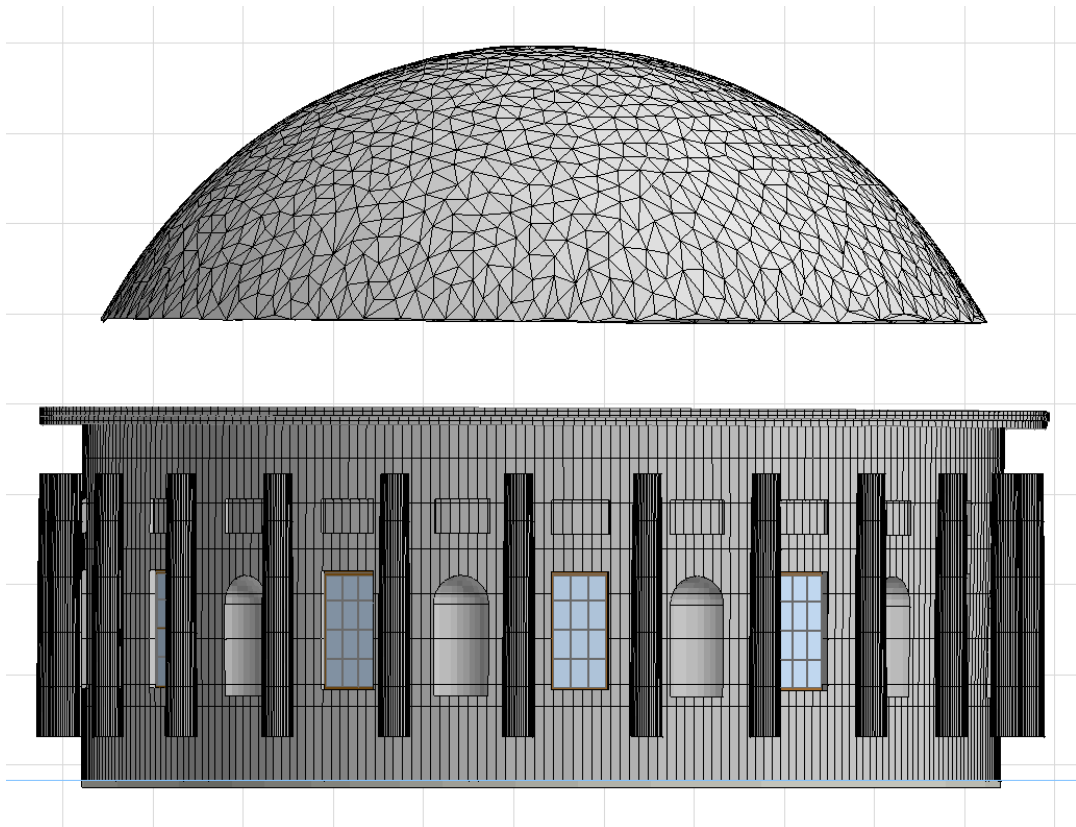


Figure 7.28: Vector elevation drawing automatically generated from the 3D model for the Four Courts, Dublin.

After analysing all the documentation produced it was clear that the area of the Four Courts drum that was most affected by deformation was the side facing south onto the front of the building and the street and river below. This is also consistent with the capitals as the capitals containing the most damage and decay are also in the same area facing south to the front of the building. This side would have also been most exposed during the bombardment in the Civil War.

Figure 7.29 shows graphically the extent of the deformation to the drum with areas of the drum that are leaning shown in green. Cloud Compare software was used for this analysis which compared a model representing an ideal vertical drum with the actual condition of the drum (Figure 7.29).

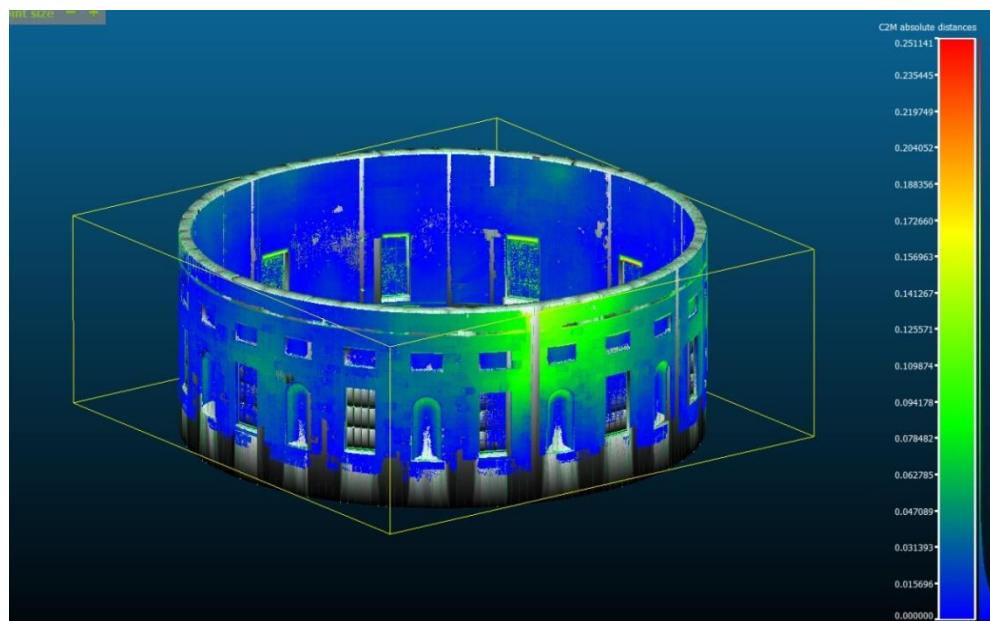


Figure 7.29: Variation between an ideal vertical drum and the actual drum which contains deformation. Areas coloured green have the highest variation while areas coloured blue have a low variation.

Other useful forms of documentation were also produced directly from scan data. This included a high definition orthographic image for the reflected internal plaster dome as viewed from the rotunda below (Figure 7.30). This can be used to identify and document the decorative plaster ceiling allowing any measurements to be taken from the orthographic image (Figure 7.30). This orthographic image of the reflected internal dome ceiling was also used for further documentation. This included marking the positions of the steel trusses used to support this internal plaster dome (Figure 7.31). Figure 7.32 shows another orthographic image for a Corinthian capital that was used to manually create a vector drawing.

### 7.3.7 Adopted Conservation Methods

Laser scanning and subsequent documentation generated from the HBIM showed the extent of the damage and decay to the Four Courts dome and drum. From this documentation, it was possible to identify the areas most affected by deformation along with identifying where steel inside the concrete dome was corroding and damaging the

concrete. Instead of trying to remove the steel, which may have required substantial demolition and reconstruction of the dome, the chosen conservation method involved installing a system known as cathodic protection. This involves inserting “sacrificial” zinc plugs into the concrete that then attract the corrosion away from the steel. As part of the conservation works, the concrete on the inside of the top dome was also resurfaced (Figure 7.34). Once all structural issues have been dealt with the conservation work required for the Corinthian capitals will also begin Figure 7.33 & Figure 7.35.

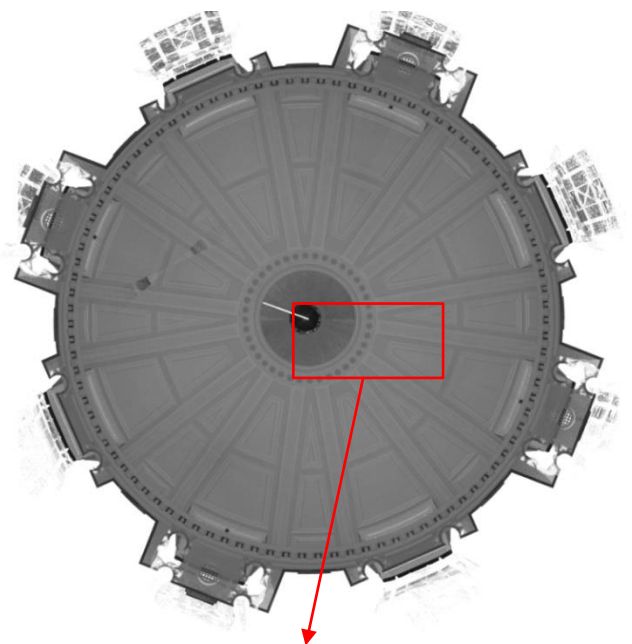
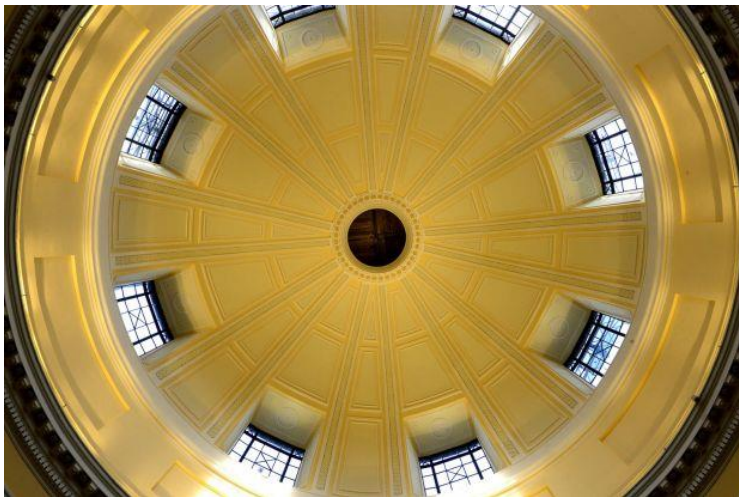


Figure 7.30: Reflected internal plaster dome of the Four Courts Rotunda viewed from below. Photo containing distortion (left) and true to scale orthographic image generated from point cloud (right).



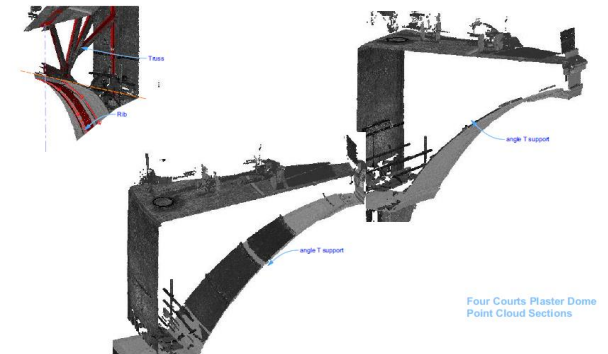
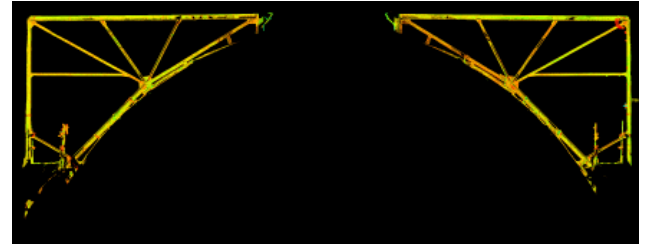
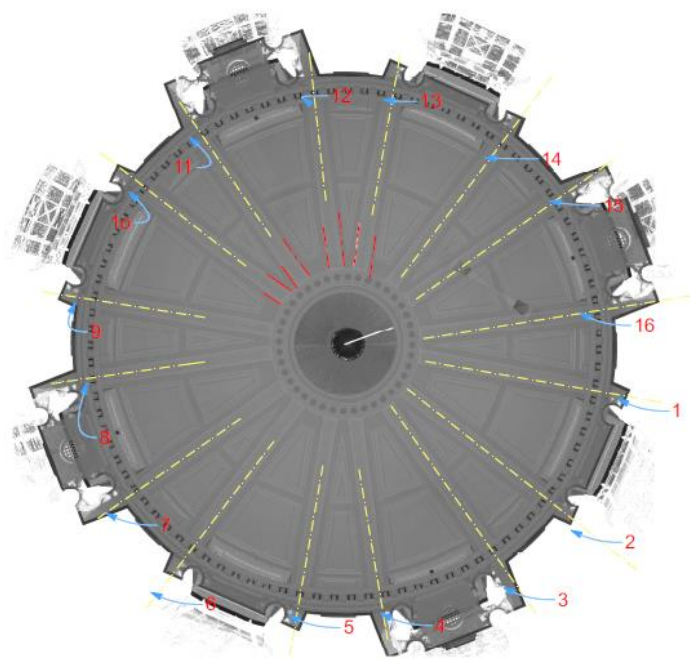


Figure 7.31: Location of steel trusses supporting the internal plaster dome is marked on the orthographic image (left). Cut-sections of same steel trusses from the point cloud are shown on the right.

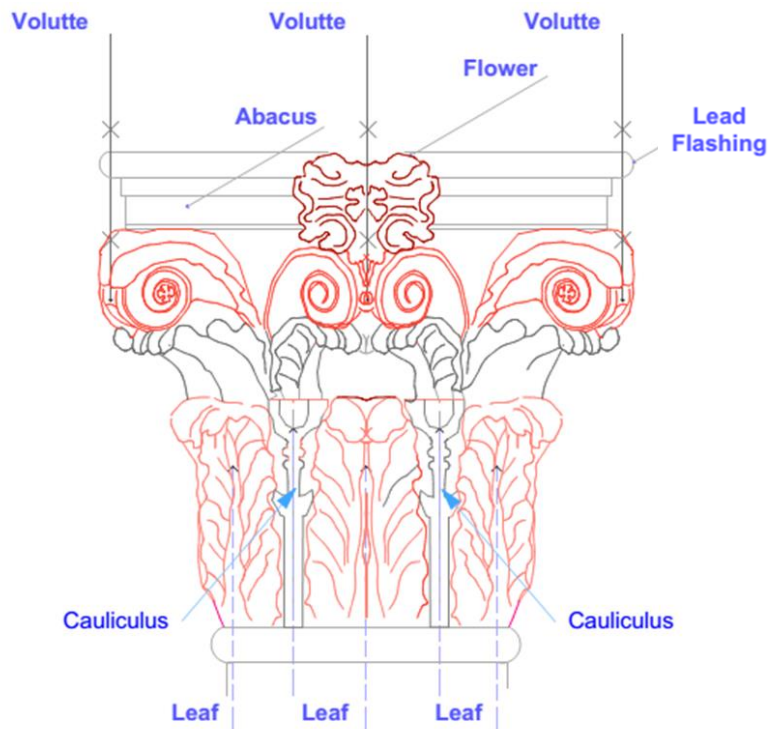


Figure 7.32: Orthographic image of a Corinthian capital from the Four Courts Dome (right). Vector documentation manually created from orthographic image is shown on the left.





Figure 7.33: Current renovation work to the Four Courts dome, drum and capitals (Byrne, 2015).

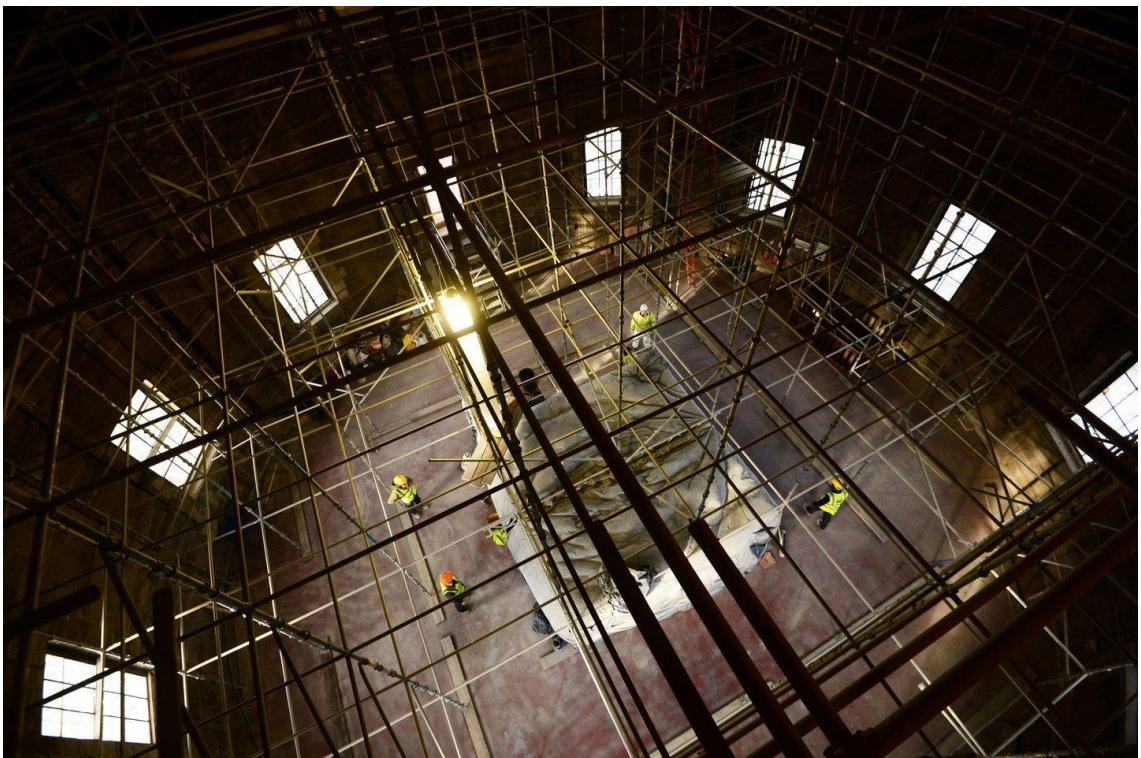


Figure 7.34: Scaffolding put in place in the top dome for resurfacing the internal concrete surface (Byrne, 2015).





Figure 7.35: Current renovation work to the Four Courts dome, drum and capitals (Byrne, 2015).

### 7.3.8 Review of Case Study

This case study showed the capabilities of the third Irregular Procedural Building Prototype for quickly generating accurate documentation to analyse a damaged structure. Using this prototype it was possible to procedurally generate irregular BIM geometry that accurately represented the drum, columns and beams. The irregular surfaces were automatically generated from cut-sections extracted from point cloud data. Once the drum wall surface was generated, the required arrangement of objects was automatically generated and then manually refined. Methods for fast and efficient graphical editing also enabled all objects to be edited simultaneously in groups or individually as required.

From the documentation produced, it was possible to identify the extent of the damage and the areas most affected by deformation. The HBIM can also be used in other dedicated software for further structural analysis such as Finite Element Modelling (FEM) software.



## **7.4 Conclusion**

This chapter described two case studies which showed the potential use of the newly developed HBIM procedural modelling prototypes. The first case study of Henrietta Street showed the benefits of using the procedural façade prototype for more efficient and accurate generation of façade models when compared to existing manual approaches. This case study also showed the potential for further applications and analysis of HBIM data which included, conservation, economic and energy analysis. The second case study of the Four Courts showed the use for the third Irregular Procedural Building prototype on a real conservation project. Without this prototype, it would not have been possible to generate accurate BIM geometry for the irregular dome and drum which contains a high level of deformation. The next chapter describes the methodology for further validation and testing used to evaluate the newly developed HBIM procedural modelling prototypes.

## **Chapter Eight: Validation and Testing**

### **8.1 Introduction**

This chapter describes the methodology undertaken to further validate and test the research hypothesis, that procedural modelling is a suitable solution for more accurate, automated and easier generation of BIM geometry from point clouds. The findings and analysis of the validation and testing are presented in Chapter 9.

In order to test the hypothesis three different tests were undertaken which included both qualitative and quantitative testing. These three tests were designed to address the three parts of the hypothesis by testing the accuracy, usability and the level of automation. The accuracy of a procedurally generated model is very important so that a model accurately depicts the as-is condition. A quantitative test was designed and implemented to validate the accuracy capabilities of the developed prototypes. This involved comparing models generated with the new prototypes to different sets of reference measurements. The deviations between a procedurally generated model and a set of reference measurements were used to quantify the accuracy capabilities of the prototypes.

The second type of testing implemented was end-user-scenario testing. The aim of this qualitative testing was to acquire feedback from end-users in industry and academia to evaluate the usefulness, efficiency, usability and accuracy capabilities of the developed prototypes.

The third type of testing involved quantifying the level of automation achieved from the developed prototypes when compared to a manual approach. This test involved measuring the time taken to generate a model with existing manual methods and the time taken to generate a similar model with the developed prototypes. The difference in

time or time-saving that is achieved from using the new approach is used to quantify the level of automation.

## **8.2 Accuracy Testing**

### **8.2.1 Accuracy Tests with Case Studies**

Two types of tests were carried out to validate the accuracy capabilities of the developed prototypes. This included a physical measurement method and a deviation analysis method (Anil et al., 2011, Anil et al., 2013). The first method, a physical measurement method involved capturing an independent set of reference measurements on a building and comparing these to virtual measurements in a procedurally generated building information model. The second test, a deviation analysis method involved analysing the patterns in the differences between a laser scan point cloud and a procedurally generated building information model. Building information models generated as part of the case studies in Chapter 7 were used for both types of accuracy test.

Data from the Henrietta Street case study was used for the first physical measurement accuracy test. This test involved capturing a set of independent reference measurements on the façade of number three Henrietta Street using a Leica TPS 1202 Total Station. This instrument has a horizontal and vertical angular accuracy of 3 seconds and a linear accuracy of 2mm + 2 parts per million (ppm) (Leica Geosystems 2009). With this type of test, it is not feasible to physically measure every possible location so only a sample of measurements are captured and used. A random sample of defined points on the building façade were measured remotely with the Total Station using a reflectorless mode (Figure 8.1). This sample included 28 points in total which were recorded from two positions to ensure accurate results. This resulted in a set of x, y and z coordinates for each measured point (Figure 8.1). These reference points were then compared to a

building information model for the same façade which was generated with the first Procedural Façade plug-in. The same set of points measured using a Total Station were measured on the generated virtual building information model (Figure 8.2). Statistical analysis could then be carried out to assess the differences between the physical and virtual measurements.

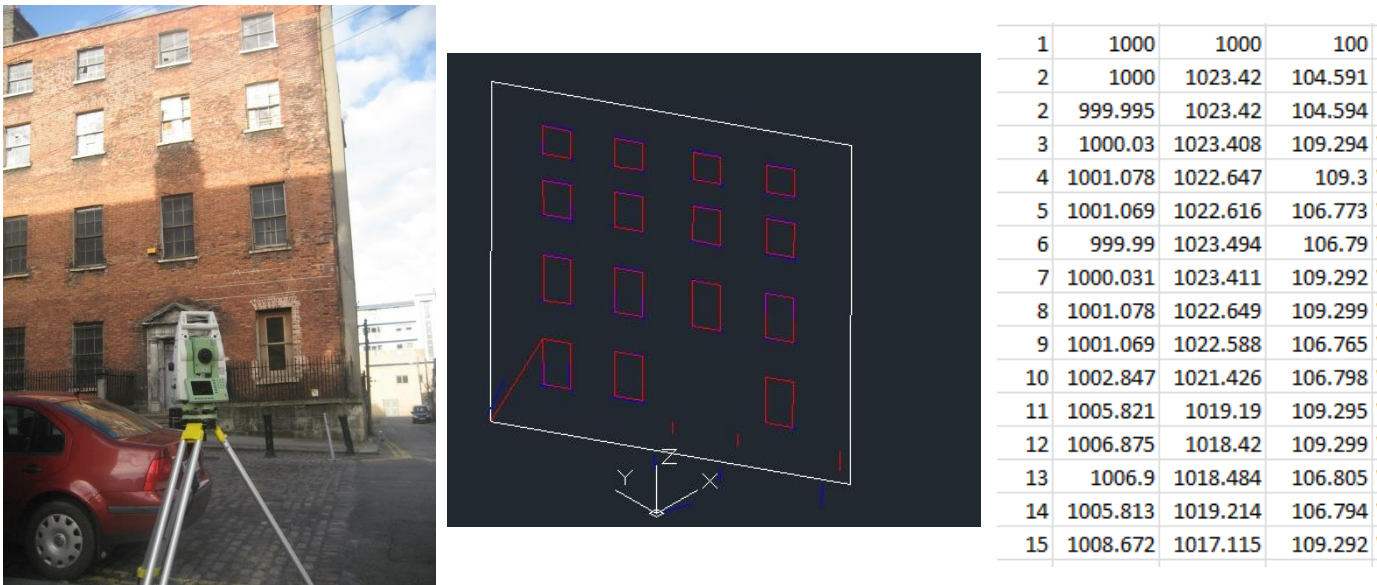


Figure 8.1: Total Station used to capture reference measurements (left), measurements viewed in AutoCAD software (middle) and coordinates viewed in excel (right).

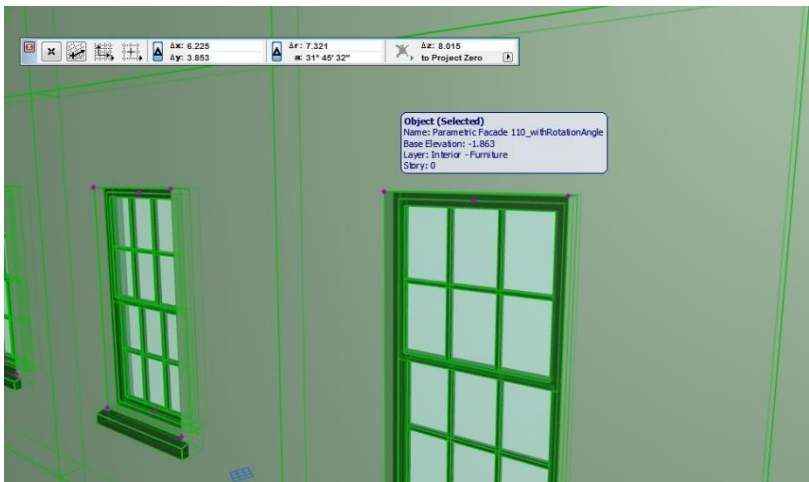


Figure 8.2: Measuring coordinates from a procedurally generated virtual façade model within the ArchiCAD software.

A limitation of the physical measurement methodology for this accuracy assessment was that the recorded Total Station measurements were not in the same coordinate reference system as the procedurally generated façade model. This was due to the fact that the façade model was generated from a laser scan point cloud which was defined in a different coordinate reference system to the Total Station survey. This required a further processing step of aligning the total station points to the same coordinate reference system as the building information model. A best-fit alignment was carried out for this using AutoCAD software. Once both datasets were in the same coordinate reference system it was possible to compare and analyse the results. The average error, standard deviation and root-mean-squared error were used to analyse the deviations between both datasets (Equation 8.1). The mean error indicates the most common or median deviation. The standard deviation is an important indicator of the quality of the results as it shows how much the individual deviations differ from the mean. The root mean square error is another measure to show the spread of the deviations. This is calculated by squaring the residuals, averaging the squares and calculating the square root.

$\bar{x} = \frac{x_1 + x_2 + x_3 \dots x_n}{n}$	$s = \sqrt{\frac{\sum (x - \bar{x})^2}{n - 1}}$	$RMSE = \sqrt{\frac{(\bar{x})^2}{n}}$
---	---	---------------------------------------

Equation 8.1: Mean error formula, standard deviation formula and root mean square error formula.

With this first accuracy test, the discrepancy between the physical Total Station coordinates and virtual façade model coordinates contained multiple possible sources for error. This included errors in total station measurements, laser scan measurements, best-fit data alignment errors along with modelling errors. A disadvantage of the physical measurement method is that the modelling errors cannot be separated from data capture and alignment errors. Only modelling errors were required for this research as

the assessment of data capture and data alignment errors were outside of the defined scope. Another disadvantage of the physical measurement method is that it only analyses the accuracy using a sample of reference points. The results, therefore, might not be a true indication of model accuracy as critical errors may be missed if they are not measured.

Due to these limitations of the physical measurement method, a second accuracy test was undertaken using a deviation analysis method. This deviation analysis method has a number of advantages over a physical measurement method. This includes the ability to isolate modelling errors. This is achieved by using a point cloud that a model is created from as a reference for testing. The deviation analysis method also uses a full set of laser scan points to test the accuracy instead of just a sample of points. This allows the accuracy of every surface and component in a building to be analysed.

For this accuracy test, a deviation analysis method was performed using the open source Cloud Compare software. This software provides functionality for point cloud processing and includes a deviation analysis method for quality assurance. With this software, it is possible to compare two point clouds or compare a point cloud to a model. Point cloud data and procedurally generated models from the Four Courts case study were used for the deviation analysis accuracy test.

An advantage of using Cloud Compare software for deviation analysis is that the process is fully automated. The software automatically finds correspondences between a point cloud and model and then computes the distances between these correspondences. Statistical analysis is then automatically performed. Outputs from the statistical analysis include the mean error and the standard deviation between corresponding points.

Results of the deviation analysis method can also be visualised graphically by colouring a point cloud with a colour scale based on the deviations. Results can be visualised using signed or unsigned deviation values. A signed visualisation colours points according to their signed deviation values so that the direction of the deviation with respect to the surface normal can be observed. This can be used to tell if a non-flat surface is bowed inward or outward (Anil et al., 2011). An unsigned visualisation colours points according to their absolute values of the deviations. A graphical representation of the deviations is particularly useful for identifying specific locations where errors are present along with identifying the type of error. Modelling errors can include:

- missing model components
- an incorrect component type
- a component modelled in the wrong location
- a component modelled with incorrect geometry (e.g. wrong sizes, scale or orientation)

A visual assessment of the results from a deviation analysis method allows the different types of errors listed above to be identified and located.

### **8.3 End-User Scenario Testing**

End-user scenario testing involves creating test scenarios which replicate a typical end users usage of a software tool. This type of scenario testing evaluates the entire workflow of a program and can be used to return valuable feedback to validate a program. Scenario-based testing is a conventional method used in software testing to identify problems and potential for improvement with a programme (Carroll, 2000). By replicating a typical end-users usage, scenario-based testing can validate a proposed systems behaviour in relation to real world usage (Borenstein, 1998).

Heesom (2004) reports that providing academics and industry practitioner's access to prototype systems during its development stage is of much benefit. However, Boloix and Robillard, (1995) and Borenstein (1998) agree that a post-implementation user evaluation can provide more useful evidence on the overall worth of a software system and allows the overall confidence in a system to be determined. According to Boloix (1997), an end-user evaluation should test the following three aspects from a user's perspective:

1. The level with which the software complies with its requirements
2. The usability and ease of use of the software tool
3. The contribution and potential benefits afforded by the tool

A post-implementation end-user evaluation was developed for testing the HBIM procedural modelling prototypes with academic and industry practitioners. The specific objectives of this testing were to evaluate the usefulness, efficiency, usability and accuracy capabilities of developed prototypes. Similar approaches for software testing have been adopted by Moore (2013) and Murphy (2012).

### **8.3.1 Participants for End-User Test**

Suitable participants for the end-user tests were chosen from industry and academia using a judgmental, purposive sampling method. This is a type of non-probability sampling where subjects are targeted based on their experience and expertise.

In quantitative research testing, probability sampling methods are generally preferred over non-probability sampling methods. A probability sampling method involves participants being chosen based on a randomised selection process. However, this method of sampling is not possible for all types of research. In qualitative research, a non-probability sampling method is adopted when there are a limited number of people that have expertise in the area being researched. This involves a researcher targeting



specific participants based on their expertise and experience in the area being researched.

For this research, the intended end-users of the developed procedural modelling prototypes are architecture, engineering, construction and heritage professionals who require 3D models of existing buildings from 3D point clouds or other survey datasets. A non-probability sampling method was required in selecting participants for this end-user testing due to the limited number of subjects with expertise and experience in this area.

### **8.3.2 User-Evaluation Sessions**

The user-evaluation sessions comprised of four different stages. In the first stage, a short presentation was made to participants to explain the background to the project and the test procedure. During this stage users were provided with the information sheet in Appendix A. Included in this information sheet was the project background, a description of the end-user test, a confidentiality statement and contact details for the principal investigator, project supervisor and the Dublin Institute of Technology's Research Ethics Committee. Participants were also asked to sign the consent form in Appendix B before taking part in the end-user evaluation.

A demonstration of the new software tools was provided to participants in the second stage of the user-evaluation session. This involved showing participants an example of the newly implemented software prototypes for procedural modelling within the ArchiCAD BIM software.

In the third stage of the user-evaluation session participants were given the opportunity to try and test the new software tools to carry out a typical end-user scenario of the software. For this participants were provided with survey data from the two case studies

described in Chapter 7. Orthographic images from the Henrietta Street case study were provided to reconstruct façade models with the first procedural façade prototype. Cut-sections from The Four Courts cases study were also provided to reconstruct models with the second and third procedural building prototypes. Basic instructions were also provided to participants on how to use the new procedural modelling prototypes. After participants generated a model they were shown how automated documentation could be produced from these models using the ArchiCAD BIM software. Participants were also made aware of the accuracy capabilities of the procedural modelling prototypes. This included a description of the accuracy tests outlined in Section 8.2 and the results achieved from these tests which are described in Chapter 9.

In the final stage of the user-evaluation, participants were asked to complete an online questionnaire to provide feedback and to evaluate the newly developed software tools.

### **8.3.3 Design of the User-Evaluation Questionnaire**

Three different types of questionnaires were established depending on the type of user that was participating in the user-evaluation session. The three questionnaires can be seen in Appendix C, Appendix D and Appendix E. Different types of users were profiled to acquire feedback on different areas of the developed prototypes. The three types of users profiled included:

1. Expert BIM users with Scan to BIM experience.
2. Users with expertise in general 3D modelling of existing architecture.
3. Conservation Architects with expert knowledge in the requirements for conservation documentation.

The first two types of users profiled would provide valuable feedback on the usefulness, efficiency, usability and accuracy capabilities of developed prototypes. The third type of

user profiled would provide feedback on the results and outputs from the developed prototypes to assess if they meet the requirements for conservation documentation.

A web-based questionnaire was designed and implemented for the three types of users using a Google Drive Form. A web-based questionnaire facilitates easy distribution and provides greater flexibility to respondents. This helps respondents to provide the highest standard of feedback at a convenient time which is not restricted to the user-evaluation session.

All three questionnaire types were designed using a common structure with different specific questions targeting the different types of user and the desired areas of feedback. The first section of each questionnaire included questions to determine the background of the participant and their level of experience and expertise in their area. In this section, questionnaires one and two also included questions on participant's preferred software and survey data for modelling existing buildings. This section also included questions about the typical accuracy requirements for projects undertaken by the participant. These questions were important to assess the current requirements of end-users in different professions. Specific multiple choice answers were provided for questions in this section with an option for adding alternative answers that were not already listed.

The second section of each questionnaire was used to assess the participant's satisfaction with current software tools for modelling or documenting existing buildings. These questions were important to identify if there is a current need in industry or academia for new solutions to model or document existing buildings. For this part of the questionnaire, a five category balanced Likert scale was used to measure the level of agreement or disagreement with a statement or question (Moore, 2013). Using a rating scale such as the Likert scale enables respondent's to express the strength of their opinion on a particular topic or question (Moore, 2013).

The final section of each questionnaire was used to gain specific feedback on the new procedural modelling prototypes. In questionnaires one and two the questions were focused on the usability, efficiency, usefulness and accuracy capabilities of the prototypes. In questionnaire three this section was focused on assessing the usefulness of deliverables from the prototypes. This was very important to establish if the results from the prototypes are suitable for their intended purpose in providing accurate conservation documentation for existing buildings. Again, a five category balanced Likert scale was used to measure the level of agreement or disagreement with a statement or question in this section.

All three questionnaires also included a final section for participants to provide further feedback in a textbox. This section was optional but allowed participants to provide further details in relation to limitations or suggestions for ways to improve the current procedural modelling prototypes. This enabled participants to provide feedback that is not restricted by the questions asked in the questionnaire.

## **8.4 Level of Automation**

The final test used to validate the research hypothesis involved quantifying the level of automation achieved from the HBIM procedural modelling prototypes.

### **8.4.1 Automated Stages in Workflow**

A complete scan to BIM workflow involves the following stages:

1. Data capture
2. Point cloud processing
3. Geometric modelling:
  - a. Generation of individual components
  - b. Combining individual components to create a larger model

#### c. Refining components to specific survey data

With the development of terrestrial laser scanning equipment and point cloud processing software, the first two stages of a scan to BIM workflow have become highly automated. However, the final stage of geometric modelling is still predominantly a manual process. The use of procedural modelling in a scan to BIM workflow has the potential to automate parts of the final geometric modelling stage.

The HBIM procedural modelling prototypes designed for this research allow individual parametric components to be automatically generated and automatically combined to create many different building arrangements. This greatly speeds up the manual process of generating and combining individual model components to create a building information model. Procedural modelling does not automate the complete modelling process though as parameters of a procedurally generated model still need to be refined to specific survey data. The encoding of architectural rules and proportions into procedural modelling rules helps to reduce the amount of further manual editing that is required. The ability to transfer survey data such as building footprints or cut-sections directly into a procedural modelling rule also greatly reduces the amount of further editing required.

In order to quantify how much of the geometric modelling stage can be automated with procedural modelling, a test was undertaken using the newly developed prototypes. This test compared the time taken for the geometric modelling stage with both a manual workflow and a procedural modelling workflow.

#### **8.4.2 Test Scenario**

In order to carry out the test, a scenario was developed which involved recreating a building information model from survey data. Survey data from the Four Courts case

study was used for this test scenario. The test scenario involved modelling the drum walls, windows and niches of the Four Courts from a series of cut-sections through the point cloud. This test scenario was completed with both a manual workflow and a procedural modelling workflow. The time taken for each workflow was recorded and then compared.

The test scenario was first completed using the third “Irregular Procedural Building” prototype. With this prototype, the polygon cut-sections for the drum were selected and the procedural modelling prototype was used to automatically generate the irregular wall object. Next, parameters of the procedural model were altered to automatically generate the required objects on the wall. This included altering parameters to generate a single floor with alternating window and niche objects. To generate these objects the number of tiles on the floor was set to twenty-four and the two types of alternating tile objects were selected from a list of available objects. The procedural modelling rules then automatically generated each object which was equally positioned around the circular wall. After the required geometry was generated the final stage involved refining the geometry based on the survey data. With the procedural modelling prototype, it was possible to simultaneously rotate all objects into position in plan view based on the cut-sections from the point cloud.

Next, the test scenario was repeated to generate a similar model using a manual workflow with the existing tools available in the ArchiCAD BIM software. First, the wall objects were placed based on survey data. The wall objects were placed at once using the ArchiCAD magic wand tool which creates a wall from a selected polygon outline. A limitation of the existing ArchiCAD tools is that it is not possible to generate a non-vertical circular wall object. For this reason, the wall had to be represented as a perfectly vertical wall even though the actual wall contains deformation and warping at

different locations. In contrast, the procedural modelling workflow enabled the true condition of the wall to be generated based on cut-sections at different heights. The final part of the manual workflow involved adding library objects for the windows and niches. The inbuilt ArchiCAD library already has the required window and niche objects. In contrast to the procedural modelling workflow, the manual workflow required each object to be manually added to the wall object and positioned based on survey data. This included manually adding twelve window objects, twelve arch-top niche objects and twenty-four rectangular niche objects which are positioned above each window and arch-top niche.

### 8.4.3 Quantifying the Improved Efficiency

After the test scenario was completed for both the procedural and manual workflows, it was then possible to compare and analyse the time taken to generate both models. In order to quantify the improved efficiency, the time saved using the procedural modelling prototype was expressed as a percentage of the time taken using an existing manual workflow (Equation 8.2).

$$\text{Percentage Decrease in Time} = \frac{(time_{\text{manual}} - time_{\text{automated}})}{time_{\text{manual}}} \times 100$$

Equation 8.2: Formula to calculate time-saving expressed as a percentage of the original time.

Using the formula in Equation 8.2 it was possible to calculate the percentage that a procedural modelling workflow is faster than a manual modelling workflow. The percentage change between two numbers depends on whether there is a reduction in value or an increase. A reduction in value will always result in a higher percentage change than an increase. That's because the higher the base, the lower the percentage for any given change.

Along with comparing the difference in time between both workflows, the accuracy of the two resulting models was also assessed. This was important to establish if the resulting models were comparable. The methodology for a deviation analysis test described in Section 8.2.1 was used to validate the accuracy of the models produced by manual and procedural modelling workflows. To quantify the difference in accuracy between both models, the accuracy of the procedurally generated model was expressed as a percentage of the accuracy of the manually generated model (Equation 8.3).

$$\text{Percentage difference in accuracy} = \frac{(\text{accuracy}_{\text{manual}} - \text{accuracy}_{\text{automated}})}{\text{accuracy}_{\text{manual}}} \times 100$$

Equation 8.3: Formula to calculate percentage difference in accuracy between both workflows

Using the formula in Equation 8.3 it was possible to calculate the percentage difference in accuracy between a procedural modelling workflow and a manual workflow.

The complete test scenario to quantify the level of automation has been recorded using Camtasia screen recording software and can be viewed from the following link.

**[https://youtu.be/ MyJ-c2ceTQ](https://youtu.be/MyJ-c2ceTQ)**

For display purposes, the speed of the video has been increased to show both the manual and procedural modelling workflows. The results of this test for the level of automation are presented and discussed in Chapter 9.

## **8.5 Conclusion**

This chapter presented a methodology for validating the research hypothesis using three different tests. This included an accuracy test, an end-user scenario test and a test to quantify the level of automation.

The methodology for validating the accuracy capabilities of the prototypes involved two separate tests. This included a physical measurement method and a deviation analysis



method. The benefits and limitations of both methods were presented and it was established that a deviation analysis test was the optimal method for testing the accuracy capabilities of the prototypes.

A methodology was developed for an end-user scenario test in order to acquire feedback from academic and industry practitioners. This feedback was used to evaluate the usefulness, efficiency, usability and accuracy capabilities of developed prototypes. The end-user test was also used to assess if the deliverables from the prototypes were fit for purpose in producing conservation documentation. A user-evaluation session was designed to allow participants to test the new software tools and then provide feedback by completing a web-based questionnaire. Three specific questionnaires were designed which targeted different types of users. This included expert BIM users, users with general experience in 3D modelling and conservation architects who could provide feedback on the quality and suitability of the resulting models and documentation.

The final test described in this chapter was a test to quantify the level of automation achieved from the HBIM procedural modelling prototypes. A test scenario was designed for this test which involved creating a Building Information Model from survey data using both a manual and a procedural modelling workflow. After the test scenario was completed for both workflows, the time taken to create both models was analysed and compared. The accuracy of both models was also compared to establish the quality of both models. After collecting and analysing the data for this test it was possible to determine the percentage time difference and the percentage accuracy difference from both manual and procedural modelling workflows. The next chapter in Part IV of this dissertation describes the results and analysis of the three tests described in this chapter.

# **Part IV**

## **RESULTS AND CONCLUSIONS**

**Part IV Summary:**

The final part of this thesis, Part IV, contains two chapters which describe the findings, analysis and conclusions of the research.

## **Chapter Nine: Research Findings and Analysis**

### **9.1 Introduction**

The previous sections, *Part II* and *Part III*, described the methodology for designing, implementing and testing three new prototypes that provide procedural modelling tools for reconstructing BIM geometry from point clouds. This section, *Part IV* includes two chapters that describe the research findings, analysis and conclusions of the research.

In this first chapter of *Part IV*, the research findings and analysis from the three tests described in Chapter 8 are presented. This includes the findings and analysis of the accuracy tests, end-user testing and level of automation testing.

### **9.2 Findings of Accuracy Tests**

This section presents the findings of the two accuracy tests described in Chapter 8. The accuracy tests were used to validate the accuracy capabilities of the new procedural modelling prototypes. The results of the physical measurement method are presented in Section 9.2.1 and the results of the deviation analysis method are presented in Section 9.2.2.

#### **9.2.1 Physical Measurement Method**

The physical measurement method described in Chapter 8 involved comparing a set of physical measurements captured with a total station to a set of virtual measurements taken on a procedurally generated building façade model for number three Henrietta Street. This façade model was generated using the first procedural modelling prototype. The full results from the comparison of total station and BIM measurements can be seen in Appendix F. A summary of the results of this comparison can be seen in Table 9.1. The resulting average error, standard deviation and root mean squared error for x-

coordinates and y-coordinates are all within 0.015 metres (Table 9.1). The resulting average error, standard deviation and root mean squared error for z-coordinates show a larger error of up to 0.035 metres (Table 9.1).

Table 9.1: Deviations between total station measurements and a procedurally generated building façade model for number 3 Henrietta Street.

<b>Total Station &amp; BIM Deviations</b>			
	<b>x</b>	<b>y</b>	<b>z</b>
<b>Average Difference (m)</b>	-0.015	-0.013	-0.035
<b>Standard Deviation (m)</b>	0.012	0.012	0.028
<b>RMSE (m)</b>	0.015	0.013	0.035

As mentioned in Chapter 8, a limitation of the physical measurement method is that it is not possible to determine the cause of errors between physical and virtual measurements. This is due to the fact that there are multiple potential causes for error which include errors in total station measurements, laser scan measurements, best-fit data alignment errors along with modelling errors. It was suspected that the large error found in z-coordinate deviations may have been as a result of data capture or alignment errors as opposed to modelling errors. To verify this, the total station measurements were compared to the laser scan point cloud measurements. A summary of the results of this comparison is shown in Table 9.2. The full set of results of deviations between total station and point cloud measurements can be seen in Appendix G.

Table 9.2: Deviations between total station measurements and laser scan point cloud measurements.

<b>Total Station &amp; Point Cloud Deviations</b>			
	<b>x</b>	<b>y</b>	<b>z</b>
<b>Average Difference (m)</b>	-0.015	-0.013	-0.036
<b>Standard Deviation (m)</b>	0.017	0.018	0.026
<b>RMSE (m)</b>	0.015	0.013	0.036

The comparison between total station and point cloud measurements showed a similar pattern of error between z-coordinates (Table 9.2). This highlights a potential error in either total station measurements, laser scan measurements or best-fit alignment of total station to laser scan measurements. To further clarify this, the sample of measurements from the procedurally generated BIM façade model were compared to the sample of laser scan measurements. Table 9.3 shows a summary of these results. The full comparison between the point cloud measurements and BIM façade model measurements can be seen in Appendix H. The results of this comparison did not show the same pattern of errors between z-coordinates. The average deviations between the point cloud and BIM façade model were all within 0.012m for x, y and z-coordinates (Table 9.3). The largest standard deviation error was 0.017m for y-coordinates while the standard deviation for x and z-coordinates were 0.013m and 0.006m (Table 9.3).

Table 9.3: Deviations between laser scan point cloud and a procedurally generated building façade model for number 3 Henrietta Street.

<b>Point Cloud &amp; BIM Deviations</b>			
	<b>x</b>	<b>y</b>	<b>z</b>
<b>Average Difference (m)</b>	0.010	0.012	0.005
<b>Standard Deviation (m)</b>	0.013	0.017	0.006
<b>RMSE (m)</b>	0.010	0.012	0.005

These results in Table 9.3 showed that the previous large error in z-coordinates between the total station measurements and BIM façade model measurements were not caused by modelling errors. The resulting deviations between the point cloud and procedurally generated façade model are an indication of the high accuracy capabilities that are possible with the first procedural façade prototype. This test is however only based on a sample of measurements. The next section presents the results from a more comprehensive accuracy test using a deviation analysis method.

### 9.2.2 Deviation Analysis Method

The deviation analysis methodology which was described in Chapter 8 involved comparing the deviations between a complete point cloud and a procedurally generated Building Information Model. Cloud Compare software was used for performing deviation analysis with reference data from the Four Courts case study. The Building Information Model used in this test was generated with the third “Irregular Procedural Building” prototype.

Segmented point clouds for the parts of the Four Courts that were modelled were first imported into Cloud Compare in e57 format. These would be used as reference datasets for the accuracy test. Next, the procedurally generated BIM model was imported into Cloud Compare. Cloud Compare cannot directly import IFC files so the BIM model was first converted into an OBJ format using ArchiCAD. Figure 9.1 shows the point cloud and model used for this accuracy test.

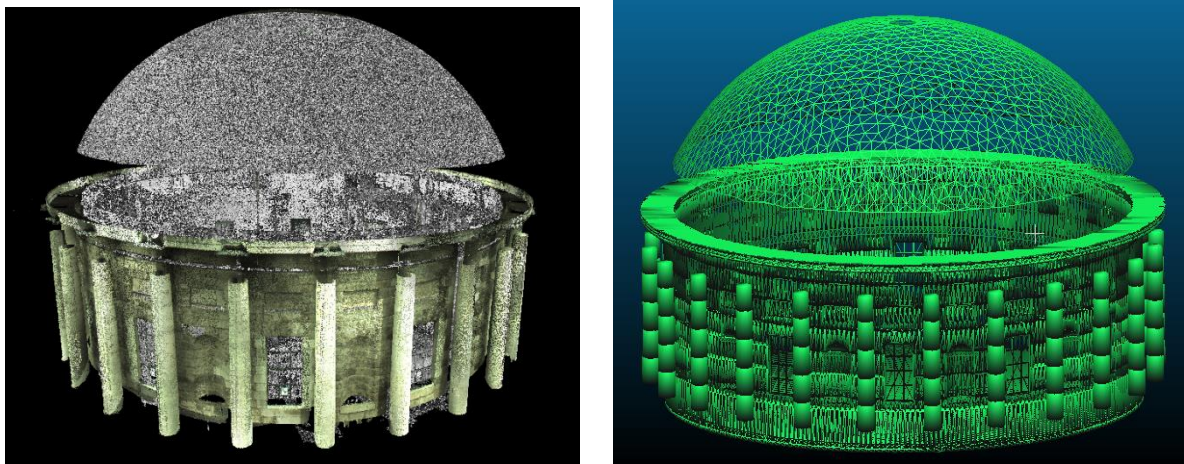


Figure 9.1: Laser scan point cloud used as a reference for accuracy test (left) and procedurally generated BIM model converted to OBJ format (right).

The deviation analysis was performed for the complete model along with individual model components in order to better track and isolate any potential errors. Deviation analysis was first performed on the drum wall which contained windows, arch-top niches and rectangular niches. A signed and absolute deviation analysis was performed to compare the procedurally generated model and point cloud. A signed deviation analysis calculates and displays the deviations as positive or negative deviations using the normal direction of model components. This can be used to assess if a non-flat surface is bowed inward or outward (Anil et al., 2011). An absolute or unsigned deviation analysis does not take into account the direction of the deviation. Instead, it calculates the absolute values of the deviations. A signed deviation analysis is better suited for visual assessments of deviations as it highlights the location and direction of deviations. An unsigned deviation analysis is better suited for statistical analysis as it provides a better indication of the mean error. This is due to the fact that a signed deviation analysis can have very low mean errors due to positive and negative deviations cancelling each other out.

Figure 9.2 shows the statistical and graphical results from an absolute deviation analysis performed on the drum walls. The resulting mean error between all points in the reference point cloud and procedurally generated model was 0.006m. The resulting standard deviation was 0.013m. These low deviation results indicate a high level of accuracy in the procedurally generated model. The mean error shows the average error at each point and the standard deviation indicates the spread of errors in relation to the mean.

The graphical results of the absolute deviation analysis showed in Figure 9.2 can be used to identify the location and type of modelling errors. Points coloured blue indicate areas of little or no deviations while points coloured green, yellow and red show areas

of larger deviations. The extent of these deviations in relation to the colour scale is identified using the key which references deviation values in metres. Points coloured red indicate the location for the largest errors of up to 0.367m. This highlights missing model components such as a stringcourse on the external drum wall and internal drain pipes. Areas coloured green indicate the location of deviations of around 0.030m due to components modelled with incorrect geometry (e.g. wrong size, scale or orientation). This can be seen on a number of arches on arch-top niches. This may indicate irregular arches that differ to the arches defined in the parametric arch-top niche object.

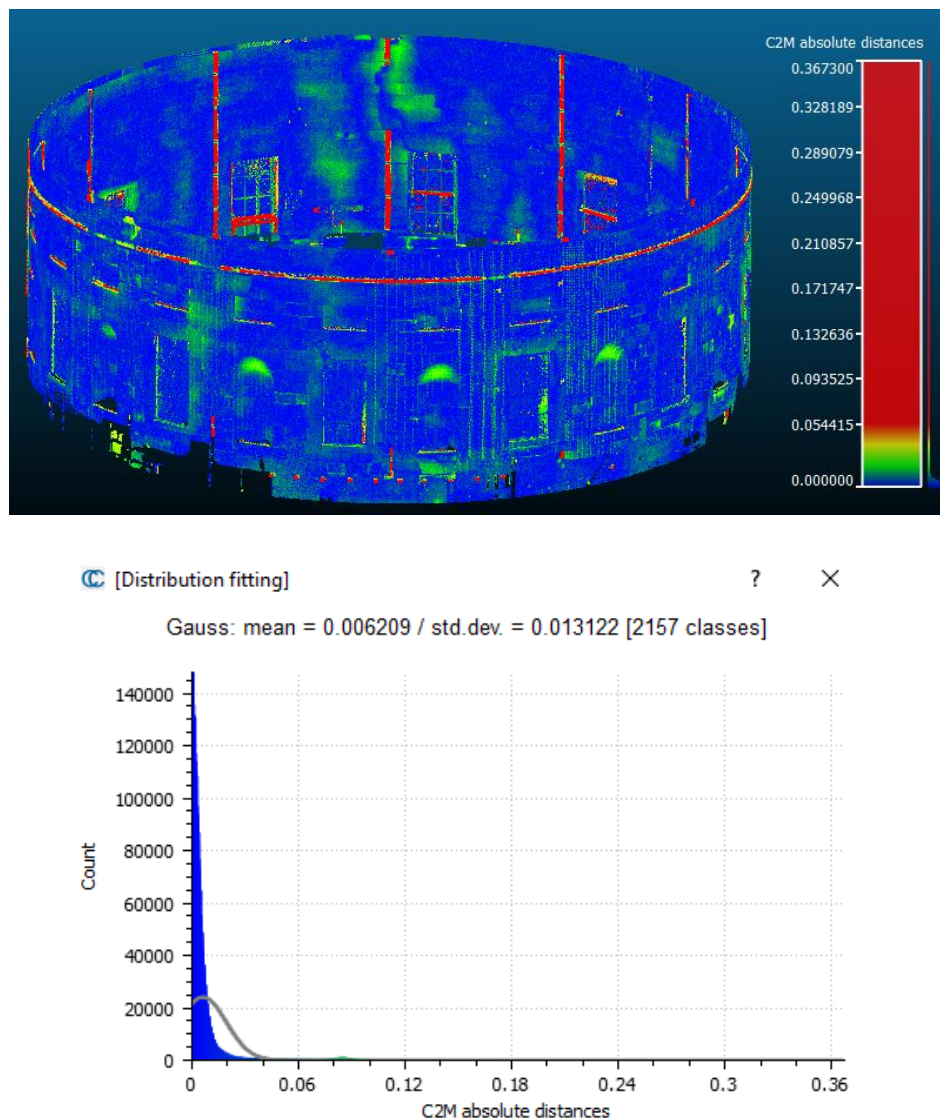


Figure 9.2: Graphical and statistical results from an absolute deviation analysis between drum wall point cloud and procedurally generated model.



Figure 9.3 shows statistical and graphical results from a signed deviation analysis of the drum point cloud and procedurally generated model. The resulting mean signed error for all points in the reference point cloud was 0.000044m and the resulting standard deviation was 0.014m. The graphical results in Figure 9.3 show similar results to the absolute deviation analysis except that it shows the direction of the deviations with respect to the surface normal from the procedural model. Deviations coloured blue such as the missing internal drain pipes show negative deviations with respect to the wall object while points coloured red such as the missing external string course show positive deviations with respect to the wall object.

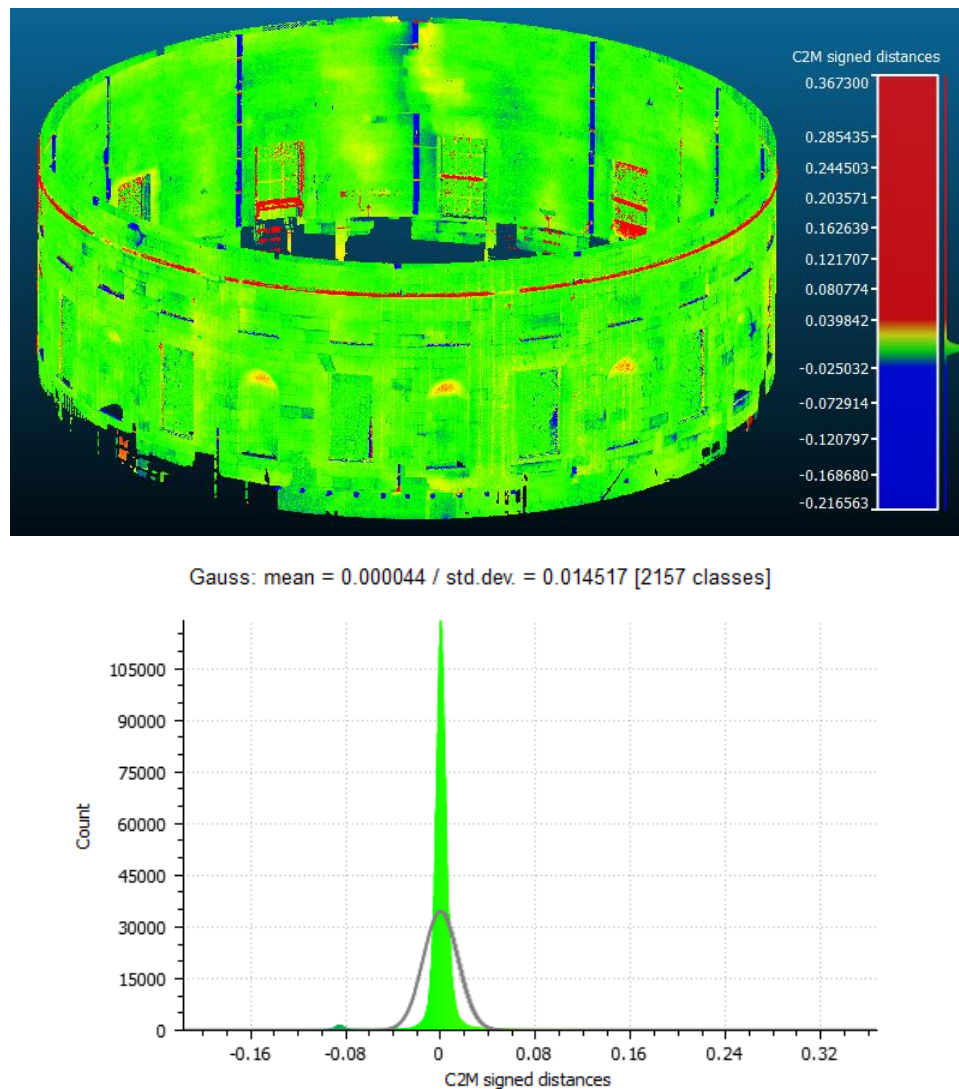


Figure 9.3: Graphical and statistical results from a signed deviation analysis between drum wall point cloud and procedurally generated model.

The next model component used in a deviation analysis was a beam encircling the drum wall (Figure 9.4). This component was generated from sections using the third “Irregular Procedural Building” prototype.



Figure 9.4: Photo (left) and procedurally generated model component for a beam encircling the drum wall.

Figure 9.5 shows the graphical and statistical results from an absolute deviation analysis between the reference point cloud and procedurally generated model component. The resulting mean deviation between all points was 0.005m. The resulting standard deviation was 0.009m. This indicates a high accuracy for this procedurally modelled component with a low level of deviation between the reference point cloud. The graphical results show the location of deviations coloured green. This highlights areas where there is a large amount of damage and deformation to stonework which was not included in the procedural model. Additional cut-sections could be used as input data to the procedural rules to more accurately model these damaged areas. Figure 9.6 shows results of a signed deviation analysis for the same beam. The resulting signed mean deviation value was 0.002m. The resulting standard deviation for the signed deviation analysis was 0.010m. The graphical results of this signed deviation show a similar pattern to the absolute results but include the direction of deviations with respect to the normal direction of the procedural model component.

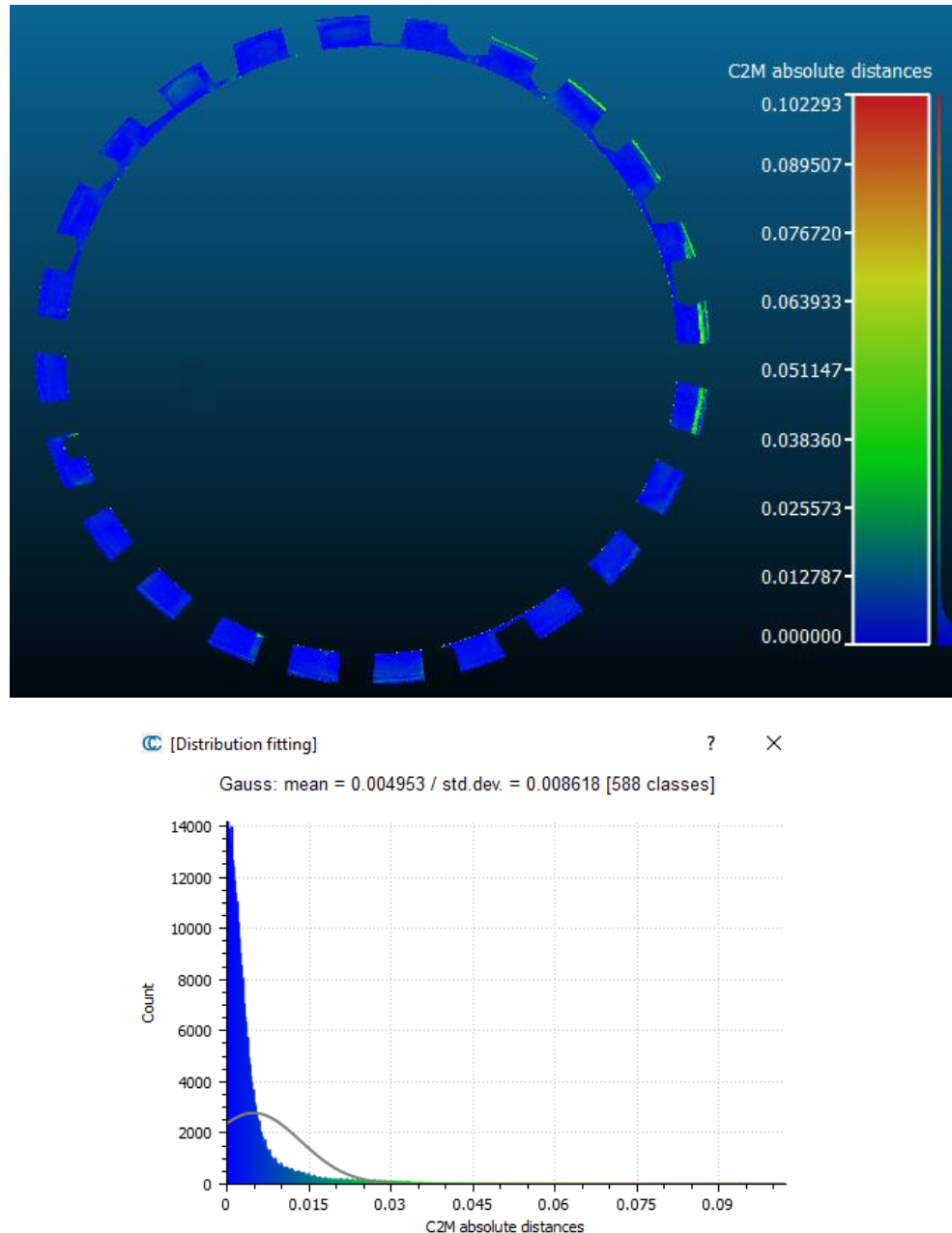


Figure 9.5: Graphical and statistical results from an absolute deviation analysis between a reference point cloud and procedurally generated model of the beam encircling the drum wall.

Deviation analysis was next performed to assess the accuracy of procedurally generated columns that surround the drum wall. Figure 9.7 shows the results from an absolute deviation analysis between the reference point cloud and procedurally generated column components. The resulting mean deviation between all points in the point cloud and procedural model was 0.003m. The resulting standard deviation was 0.005m. This

indicates a high accuracy for the procedurally modelled column components with a low level of deviation between the reference point cloud. The graphical results show the column point cloud coloured blue indicating little or no deviation between reference point cloud and procedural model.

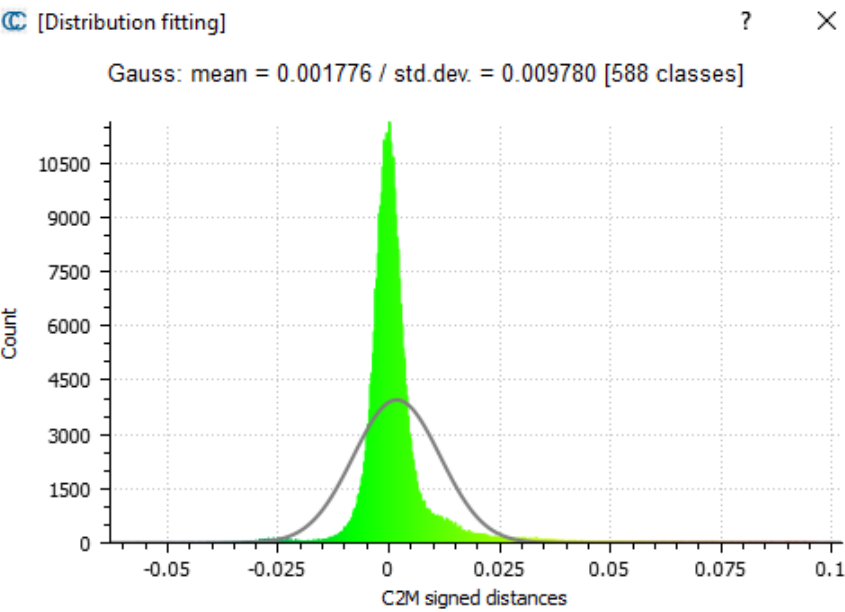
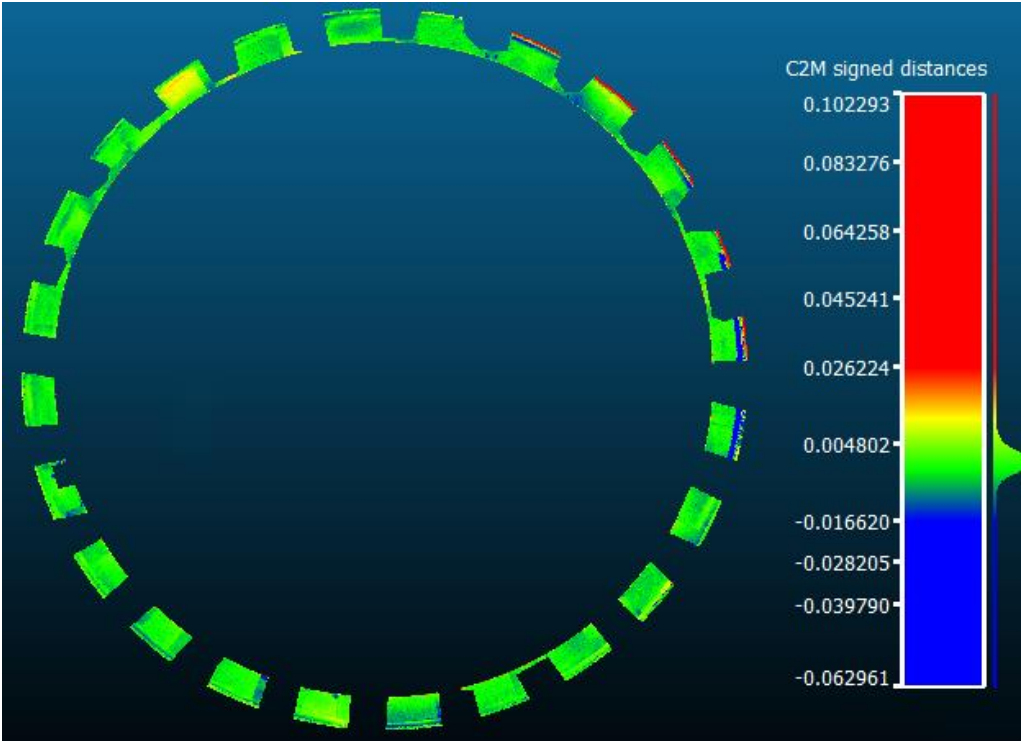


Figure 9.6:

Graphical and statistical results of a signed deviation analysis between a reference point cloud and procedurally generated model of the beam encircling the drum wall.

Figure 9.8 shows results from a signed deviation analysis for column components. The resulting mean error between the reference point cloud and procedurally generated column components was 0.001m and the standard deviation was 0.006m. Similar to the absolute deviation analysis, the signed deviation analysis results also showed a high level of accuracy between the reference point cloud and procedurally generated column components.

Finally, deviation analysis was performed on the complete model which included all procedurally modelled components. Figure 9.9 shows results from an absolute deviation analysis between the complete procedural model and reference point cloud. The resulting mean error between all points in the point cloud and the procedural model was 0.006 and the standard deviation was 0.013m. These results indicate a high level of accuracy between the complete procedural model and reference point cloud. Figure 9.10 shows the results from a signed deviation analysis between the complete procedural model and reference point cloud. Similar high results can be seen in this signed deviation analysis with a resulting mean error of 0.002 and standard deviation of 0.014m.

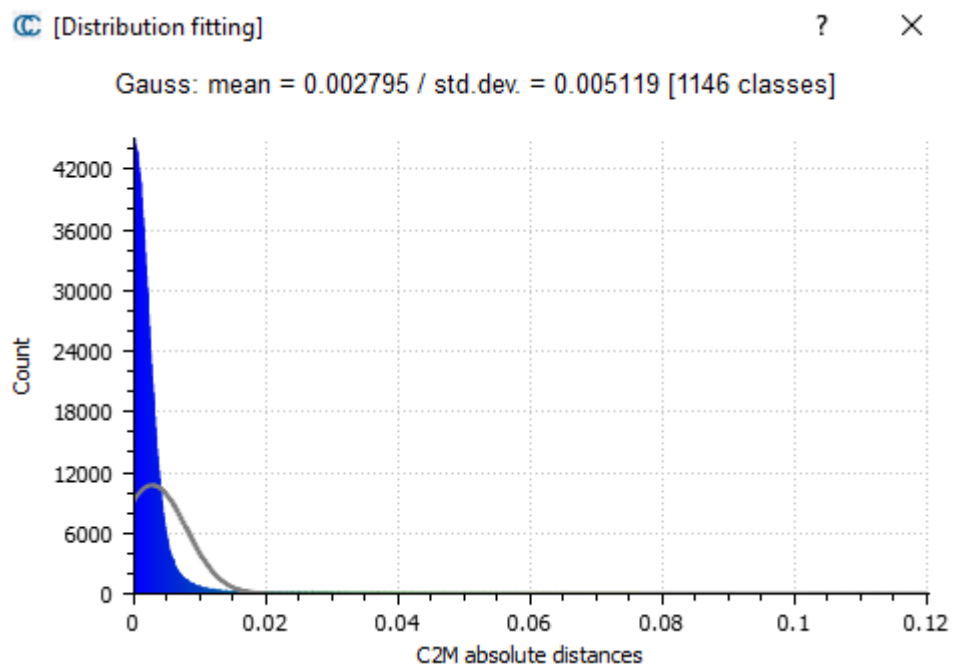
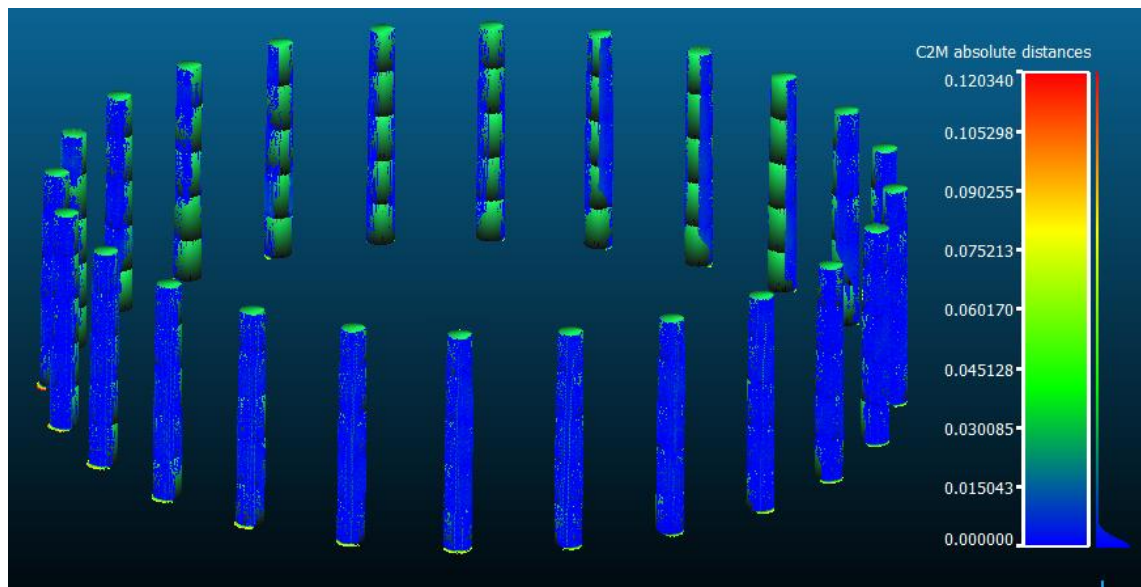


Figure 9.7: Graphical and statistical results of an absolute deviation analysis between a reference point cloud and procedurally generated model of the columns surrounding the drum wall.

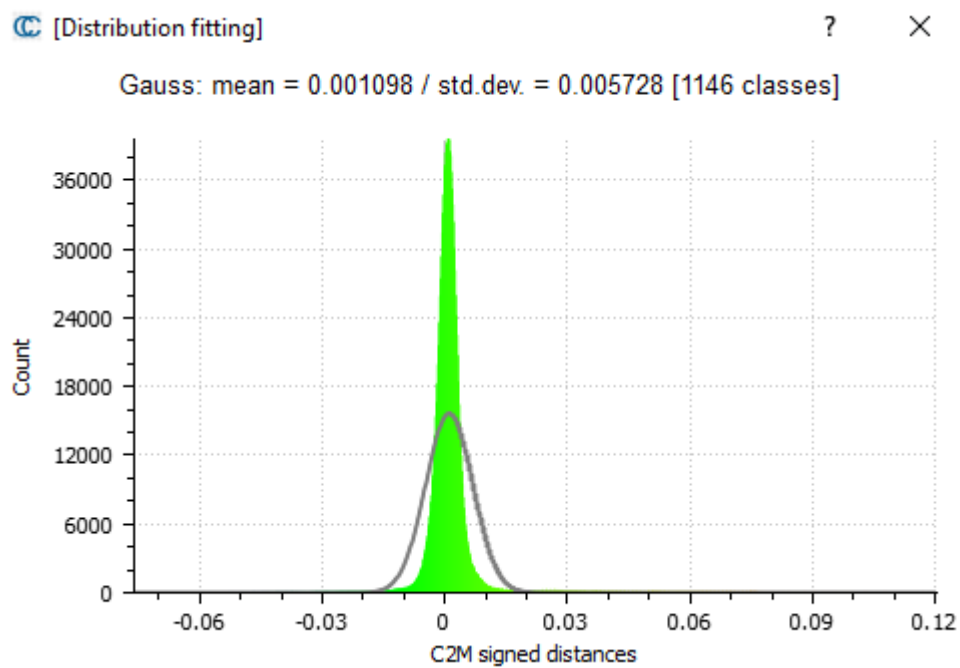
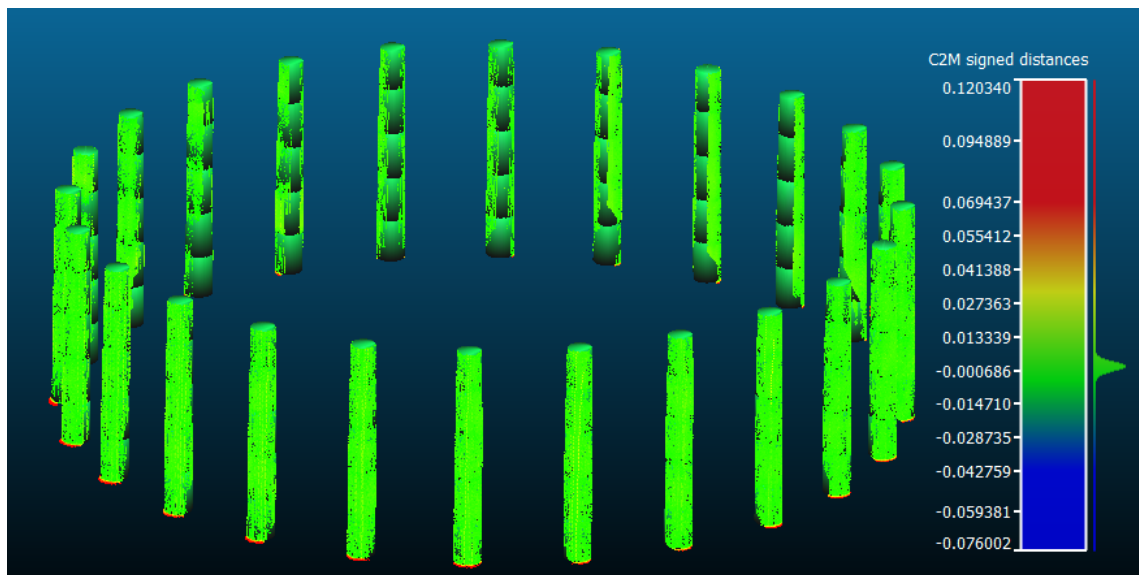


Figure 9.8: Graphical and statistical results of a signed deviation analysis between a reference point cloud and procedurally generated model of the columns surrounding the drum wall.



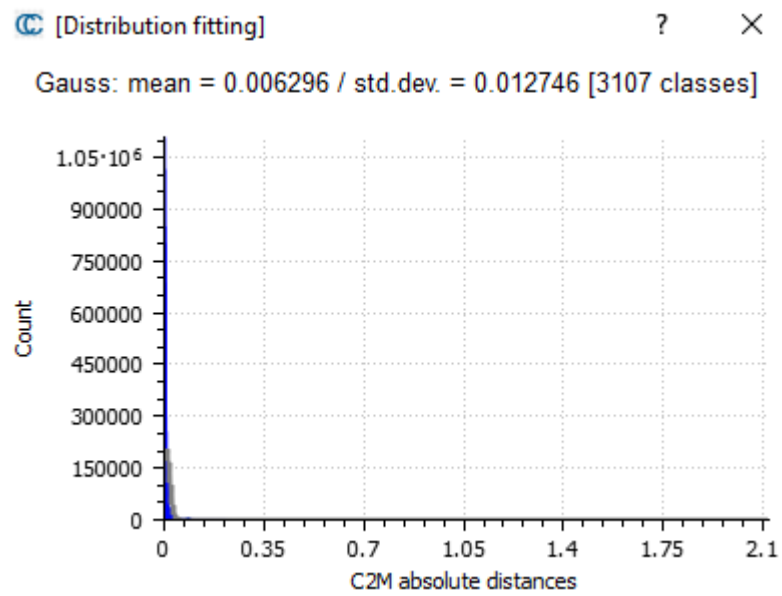
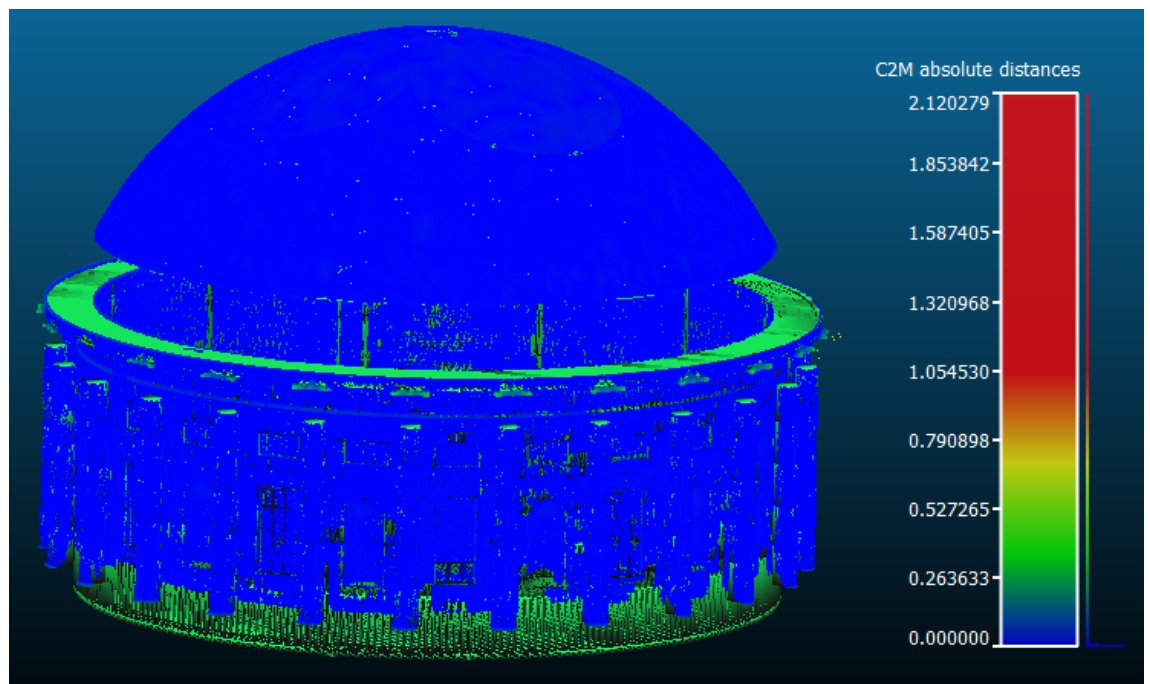
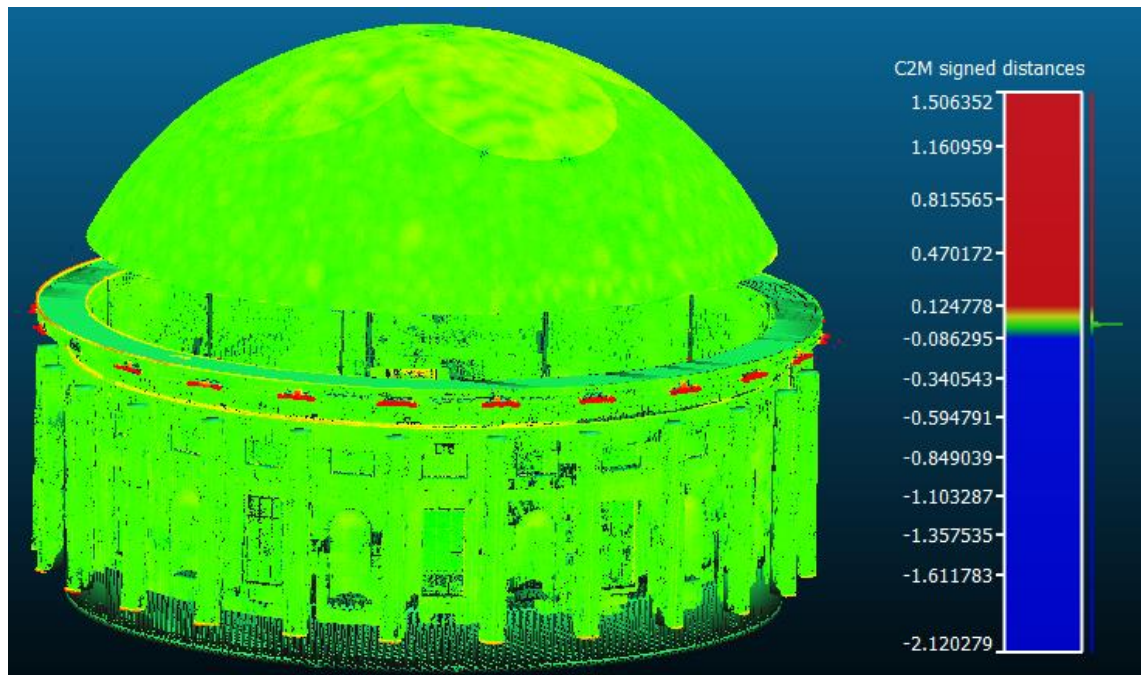


Figure 9.9: Graphical and statistical results of an absolute deviation analysis between a reference point cloud and all procedurally generated components.





© [Distribution fitting]

? ×

Gauss: mean = 0.002263 / std.dev. = 0.014035 [3107 classes]

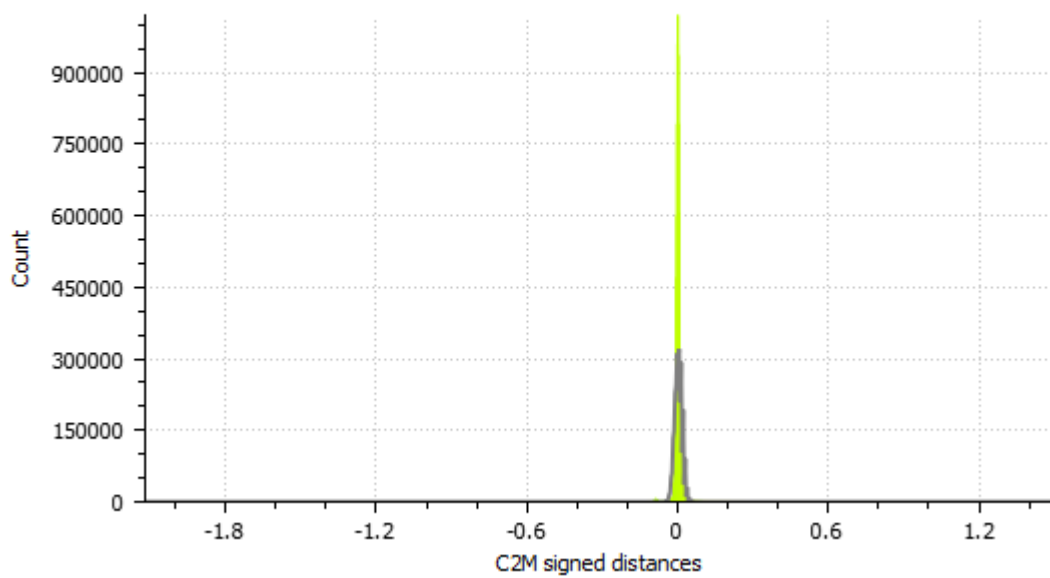


Figure 9.10: Graphical and statistical results of a signed deviation analysis between a reference point cloud and all procedurally generated components.

As an additional check on the accuracy of the procedurally generated model, cut-sections were generated from the combined point cloud and procedurally generated model. This enabled a visual assessment to be carried out on the accuracy of the procedurally generated model at sampled locations in the x, y and z-axis. Figure 9.11 shows the combined point cloud and procedurally generated BIM model used for extracting cut-sections. Figure 9.12 and Figure 9.13 show extracted cut-sections which include the procedurally generated model geometry coloured blue and the reference point cloud coloured by intensity values (yellow and orange). A visual inspection of these cut-sections indicated a good correlation between the procedurally generated model and reference point cloud.

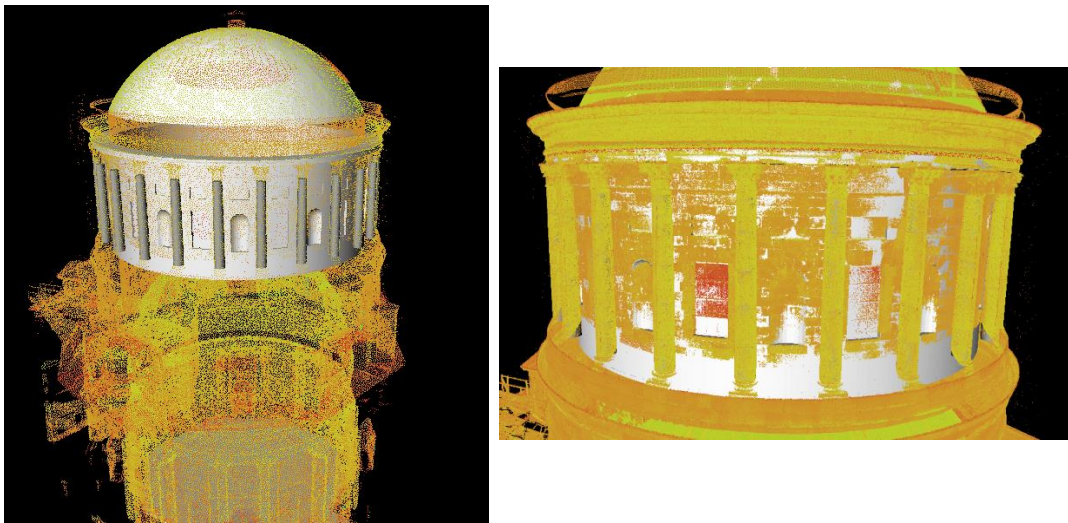


Figure 9.11: Combined point cloud and procedurally generated BIM model used for accuracy testing.

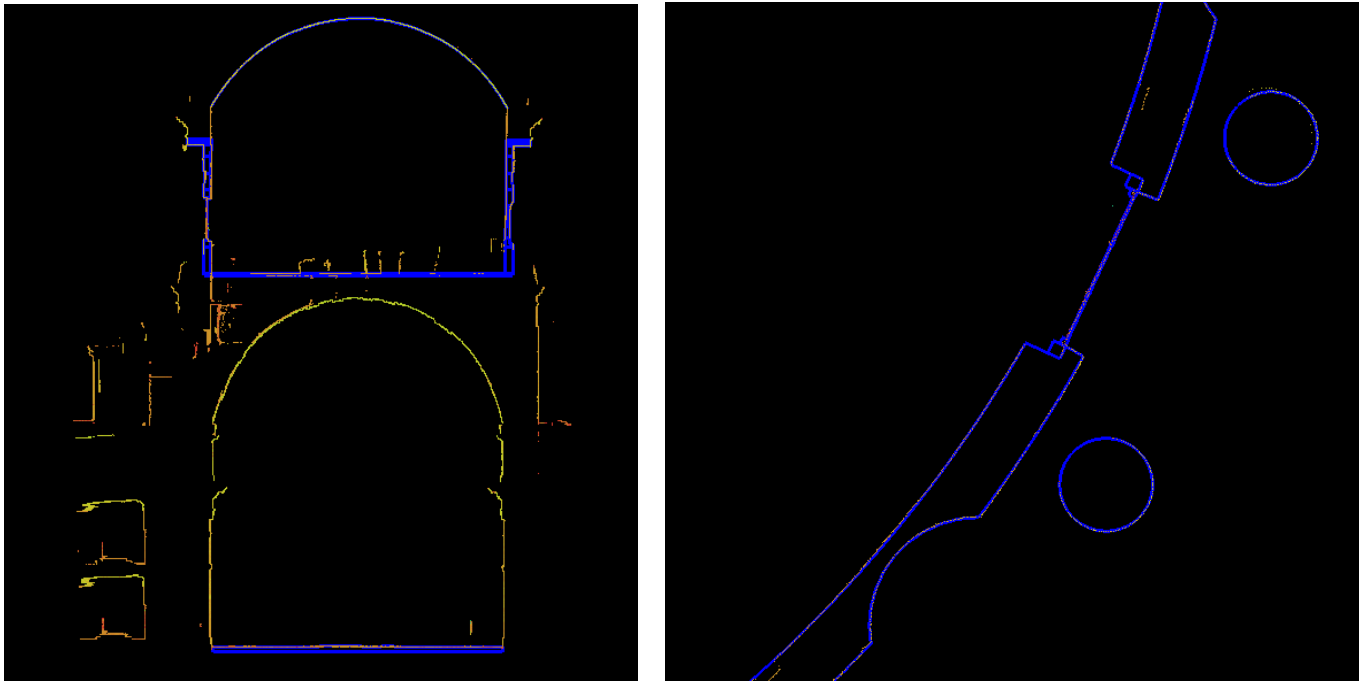


Figure 9.12: Cut-sections through point cloud (coloured yellow and orange) and generated BIM geometry (coloured blue) used for a visual accuracy assessment.



Figure 9.13: Cut-sections through point cloud (coloured yellow and orange) and generated BIM geometry (coloured blue) used for a visual accuracy assessment.

### **9.3 Findings of End-User Scenario Testing**

This section presents the results of the end-user scenario tests described in Chapter 8. The aim of this end-user test was to obtain feedback from academic and industry practitioners to evaluate the usefulness, efficiency, usability and accuracy capabilities of developed procedural HBIM prototypes. As described in Chapter 8, three different questionnaires were developed targeting different types of users. This included experienced BIM users, users with experience in general 3D modelling and conservation architects with expert knowledge in the requirements for conservation documentation. The results from these three questionnaire types are organised and presented in the sections below.

Section 9.3.1 presents the results from the first part of each questionnaire which included questions to determine the background of the participant and their level of experience in the areas relevant to this research. This section also included questions to assess end-user requirements. Section 9.3.2 presents the results of the second part of each questionnaire which was used to assess the participant's satisfaction with current software tools for modelling and documenting existing buildings. Section 9.3.3 presents the results from the final part of each questionnaire which was used to gain specific feedback on the new procedural HBIM prototypes.

#### **9.3.1 Participant Characteristics and End-User Requirements**

Six industry practitioners and six academic participants took part in the end-user test. The industry participants included three senior conservation architects working for the Office of Public Work (OPW), two senior engineers working for Headcount Engineering/BIM & Scan and one surveyor working for Survey Instrument Services (SIS). The academic participants included one full-time postdoctoral research scientist at the University College Dublin (UCD) and five full-time undergraduate students at the

Dublin Institute of Technology (DIT). The focus of the postdoctoral research scientist's work is on automatic building and object reconstruction from aerial and terrestrial laser scan data. The five undergraduate students were fourth-year students studying for a Bachelor of Science Degree in Construction Management. As part of their studies, these undergraduate students had completed modules and were experienced in 3D CAD, BIM, surveying and building maintenance and conservation.

The first question in this part of the questionnaire asked participants to identify the core activity of their organisation. The responses from participants showed a range of activities including architecture, engineering, surveying, conservation, construction management and academia (Figure 9.14). Figure 9.15 shows additional information about the test participants which include participant's organisations and the percentage of industry and academic participants.

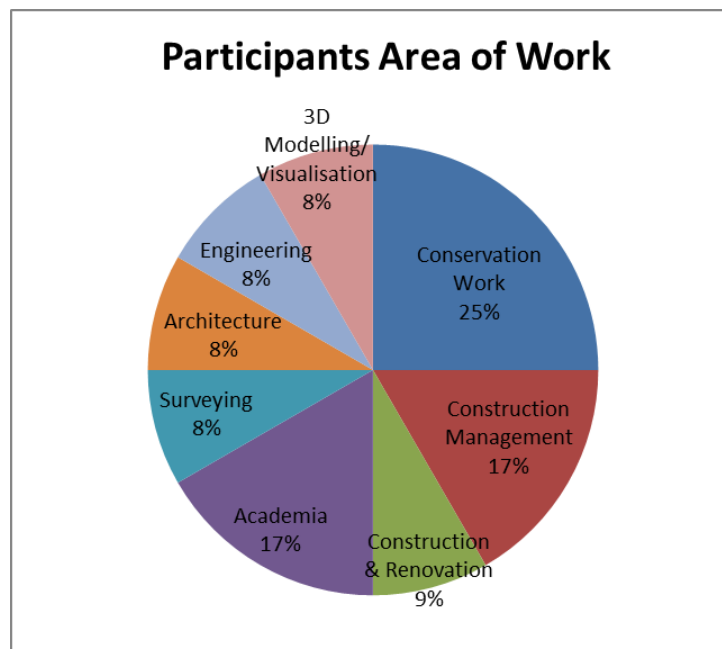


Figure 9.14: Results of end-user scenario testing - "Participants Area of Work".

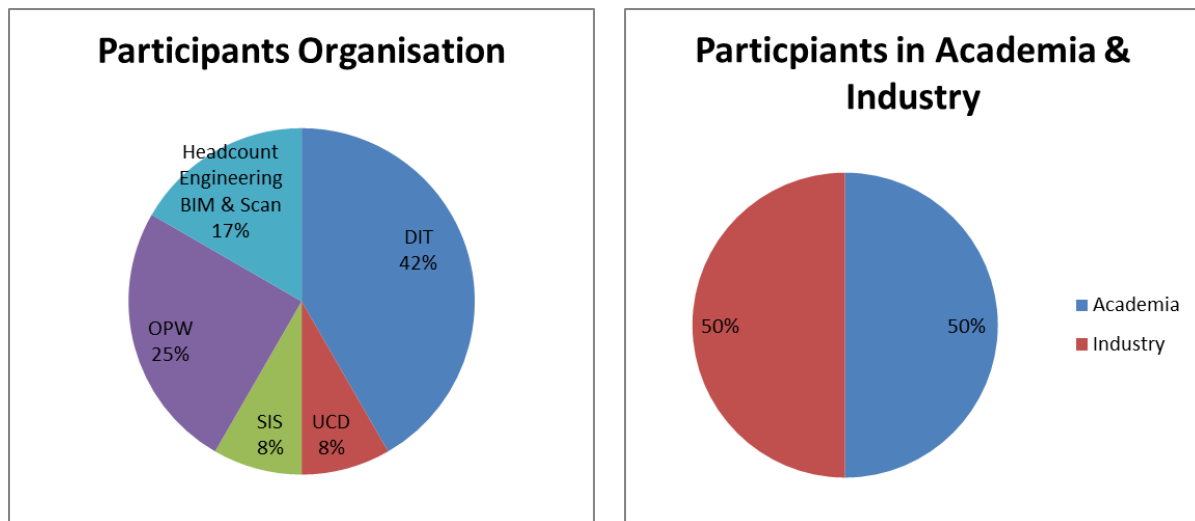


Figure 9.15: Results of end-user scenario testing - “Participants Organisation” and “Participants in Academia and Industry”.

Next participants were asked about their experience in the areas relevant to this research. Questionnaires one and two were focused on users who would create models with the new software plug-ins while questionnaire three focused on users who work with the results from the new software plug-ins. For this reason, questionnaires one and two asked participants how much experience they had modelling existing buildings while questionnaire three asked participants about their experience with conservation projects. The overall results are shown in Figure 9.16. 50% of participants had moderate experience (3 projects or more), 30% had some experience (1 – 2 projects) and 20% had frequent experience (large number of projects). These results show that the majority of the participants had a high level of experience in the specific areas required for assessing the new procedural HBIM software plug-ins.

Next participants were asked what software and what source data they mostly used for modelling existing buildings. This information was important for assessing a user’s software and data requirements. The responses from these questions are shown in Figure 9.17. The main software used by participants for modelling existing buildings was AutoCAD (42%), ArchiCAD (25%) and Revit (17%). Other software (2%)

included Cyclone, Bentley and AllPlan. The preferred source data for modelling existing buildings was laser scan data (42%), combination of methods (33%), photogrammetry (8%) and existing 2D documentation (8%). Other source data mentioned included commissioned total station and GNSS surveys (8%).

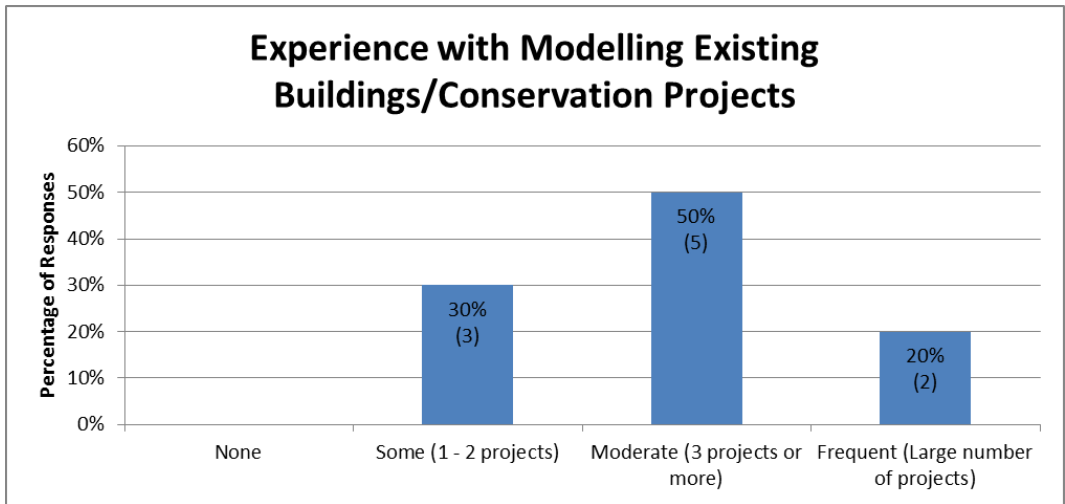


Figure 9.16: Results of end-user scenario testing – “Experience with Modelling Existing Building/Conservation Projects”.

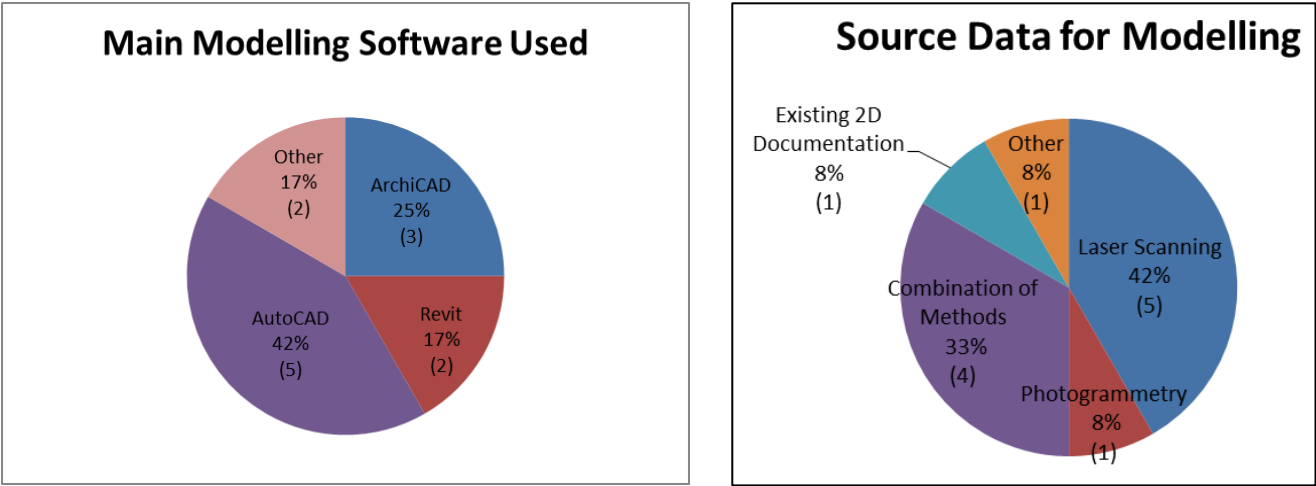


Figure 9.17: Results of end-user scenario testing – “Main Modelling Software Used” and “Preferred Source Data for Modelling Existing Buildings”.



The final question in this first section of each questionnaire was used to assess the typical accuracy requirements for projects undertaken by the participant. This was important to assess current end-user requirements. The results of this can be seen in Figure 9.18. 36% of participants required accuracies less than 1cm, 27% required accuracies less than 5mm, 18% required accuracies less than 2cm, 9% required accuracies less than 5cm and 9% stated accuracy was not important. These responses show that the majority of projects undertaken by participants require a high level of accuracy.

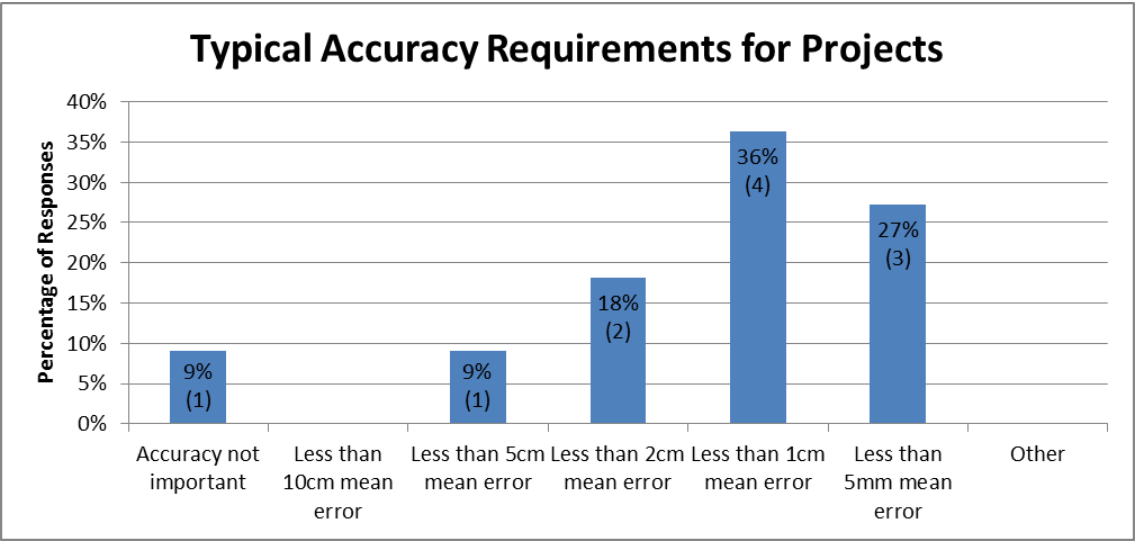


Figure 9.18: Results of end-user scenario testing – “End-User Accuracy Requirements”.

### 9.3.2 Evaluation of Current BIM software and Workflows

In the second section of each questionnaire, participants were asked to evaluate current software and workflows for modelling existing buildings. First participants were asked if current modelling software provides sufficient tools for modelling existing buildings. 56% of participants stated that the tools in current software are somewhat sufficient for modelling existing buildings. 45% of participants stated that the tools in current software are insufficient for modelling existing buildings (Figure 9.19).

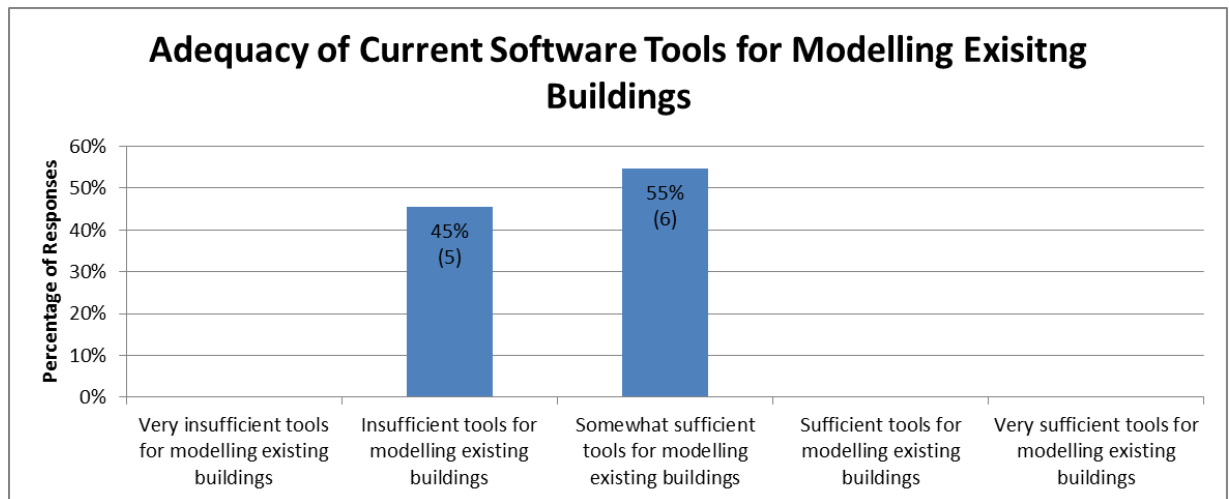


Figure 9.19: Results of end-user scenario testing – “Adequacy of” Current Software Tools for Modelling Existing Buildings”.

Next participants were asked about the efficiency of current software tools for modelling existing buildings. 40% of participants stated that current software is somewhat sufficient, 40% stated that current software is inefficient and 20% stated that current software is efficient for modelling existing buildings (Figure 9.20).

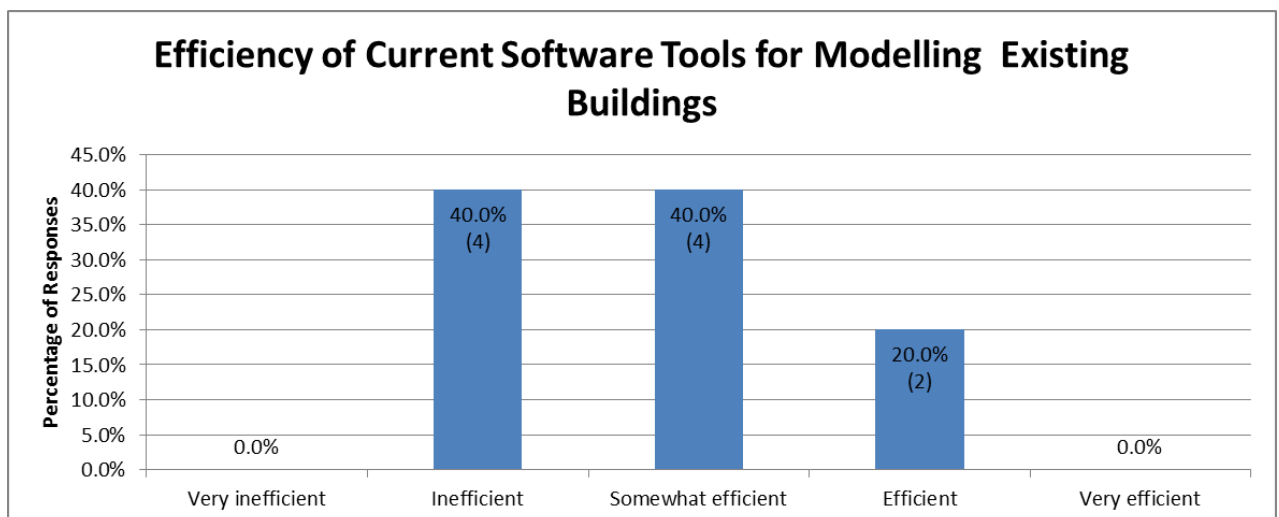


Figure 9.20: Results of end-user scenario testing – “Efficiency of” Current Software Tools for Modelling Existing Buildings”.

Participants were then asked what they find to be the greatest challenge when modelling existing buildings. 36% stated it was the difficulties in modelling the true condition of

buildings, 36% stated it was the manual creation of custom objects, 18% stated it was the lack of library objects for existing buildings and 9% stated it was working with survey data in modelling software (Figure 9.21).

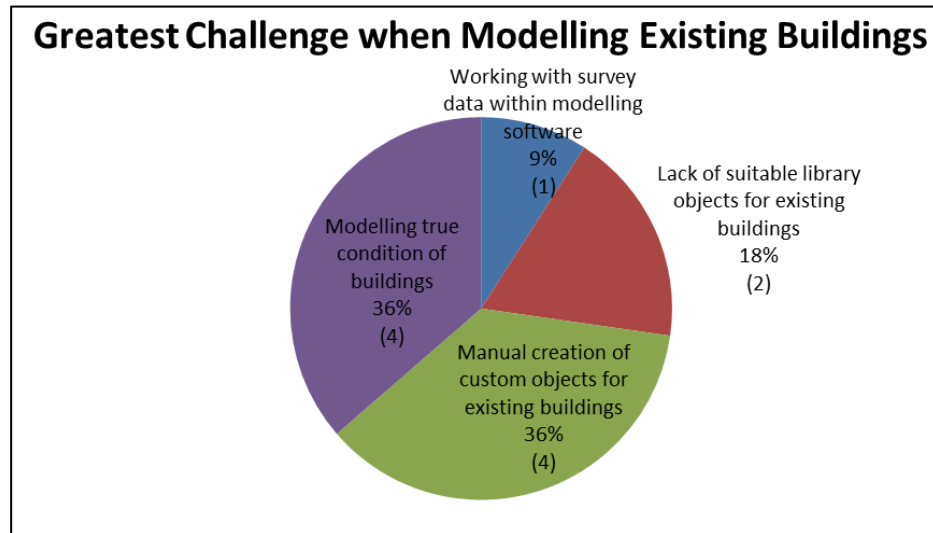


Figure 9.21: Results of end-user scenario testing – “Greatest Challenge when Modelling Existing Buildings”.

Participants were next asked if they currently use any automation in their workflows for modelling existing buildings. Participants were then asked if they think there is a need for more automation in workflows for modelling existing buildings. Figure 9.22 shows the responses from these two questions. 80% of participants currently do not use any automation in their modelling workflows and 20% stated that they currently do use automation in their modelling workflows. The types of automation used includes the use of algorithms for generating rectangular doors and windows without any architectural detail and the use of software plug-ins like Edgewise for cylindrical and planar surface detection. When asked if there is a need for more automated modelling workflows, 50% strongly agreed that there is a need for more automated workflows, 20% agreed that there is a need for more automation, 20% were neutral and 10% strongly disagreed that there is a need for more automation (Figure 9.22).

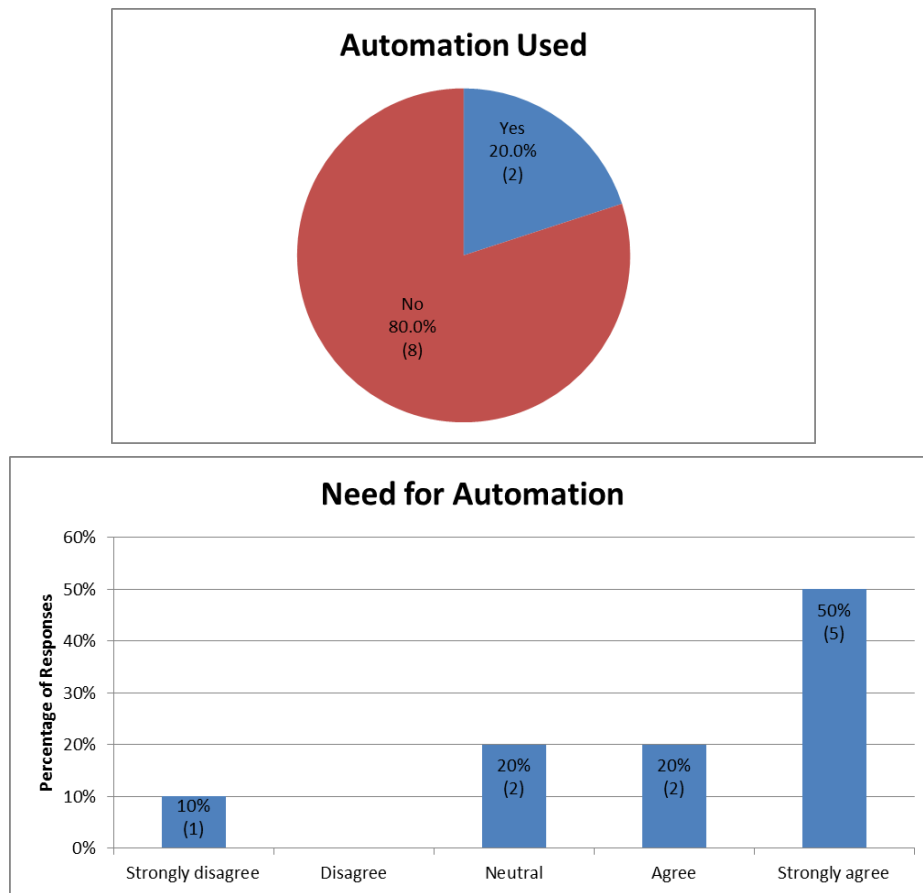


Figure 9.22: Results of end-user scenario testing – “Automation Used” and “Need for Automation”.

The final question in this section asked participants to comment on the usefulness of a procedural modelling approach for modelling existing buildings. 45% of participants stated that a procedural modelling approach would be very useful, 27% stated procedural modelling would be useful and 27% stated procedural modelling would be somewhat useful (Figure 9.23).

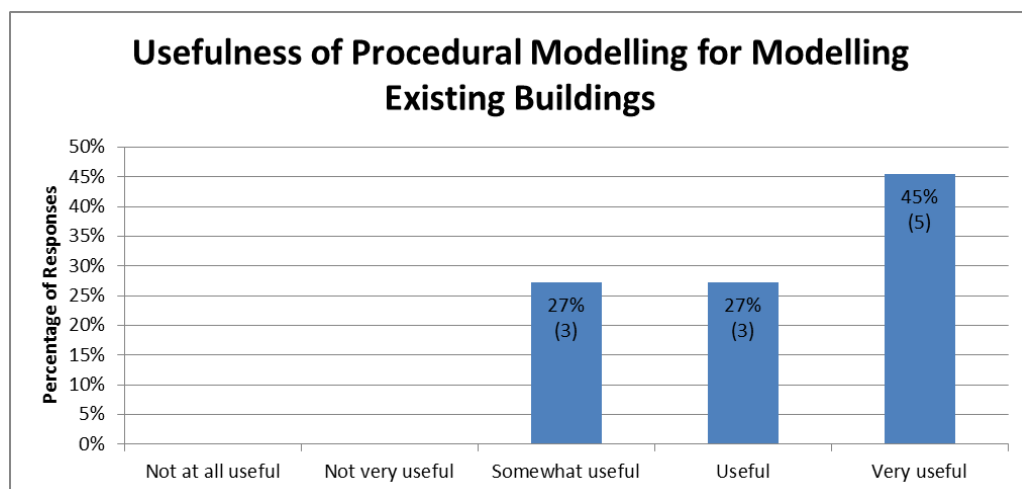


Figure 9.23: Results of end-user scenario testing – “Usefulness of Procedural Modelling for Modelling Existing Buildings”.

The responses from this second part of the questionnaire indicate that current modelling software does not have fully sufficient tools for accurately and efficiently modelling existing buildings. The majority of participants stated that there is a need for more automated workflows with better tools for accurately modelling existing buildings.

### 9.3.3 Evaluation of New Procedural HBIM Approach

In the final part of each questionnaire, participants were asked to evaluate the new procedural HBIM prototypes. Participants were first asked to rate the usefulness of the new procedural HBIM prototypes for modelling existing buildings. 73% of participants rated the new prototypes to be very useful, 9% useful, 9% somewhat useful and 9% not very useful (Figure 9.24).

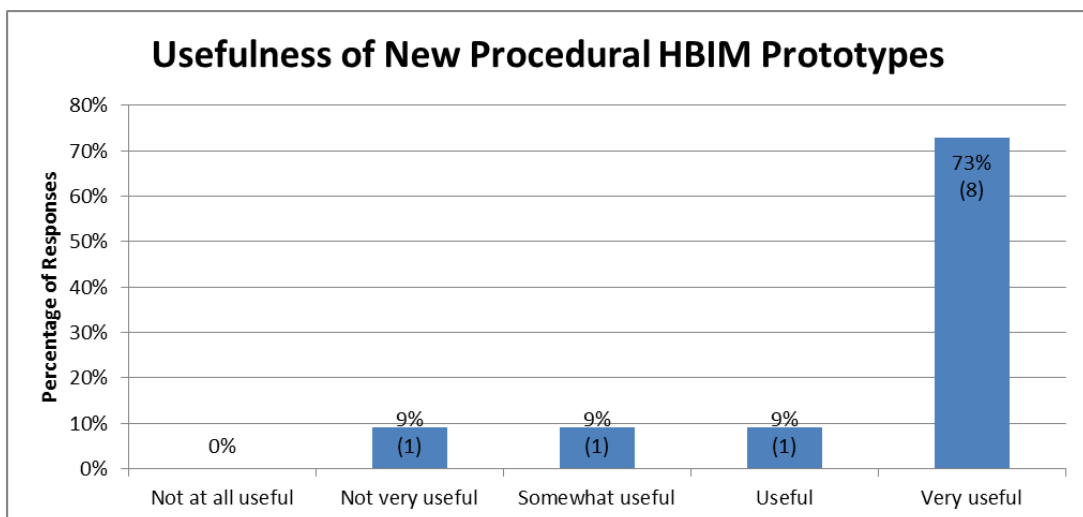


Figure 9.24: Results of end-user scenario testing – “Usefulness of New Procedural HBIM Prototypes for Modelling Existing Buildings”.

Next participants were asked what aspects of the new prototypes (if any) would they find most useful. 60% of participants responded that the automatic generation of non-vertical wall objects from sections was the most useful. 30% responded that the automatic generation of library objects on buildings was the most useful. 10% responded that group and individual graphical editing of objects was the most useful (Figure 9.25).

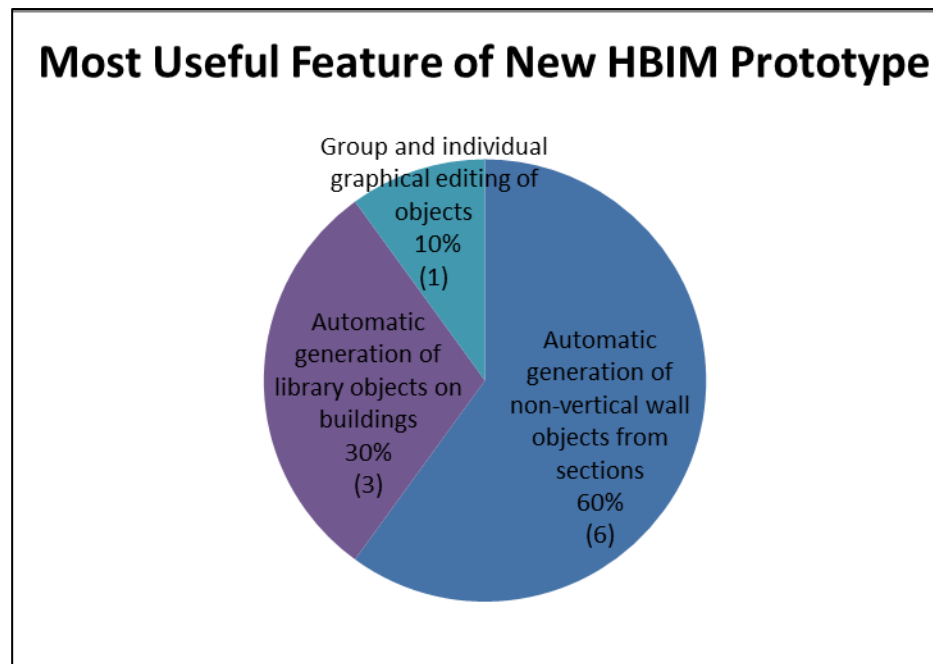


Figure 9.25: Results of end-user scenario testing – “Most Useful Feature of New Procedural HBIM Prototypes”.

Next participants were asked to rate the efficiency of modelling with the new procedural HBIM prototypes when compared to current modelling methods. 50% of respondents stated that the new procedural HBIM prototypes are much more efficient than current methods and 50% stated that the new procedural HBIM prototypes are more efficient than current methods (Figure 9.26).

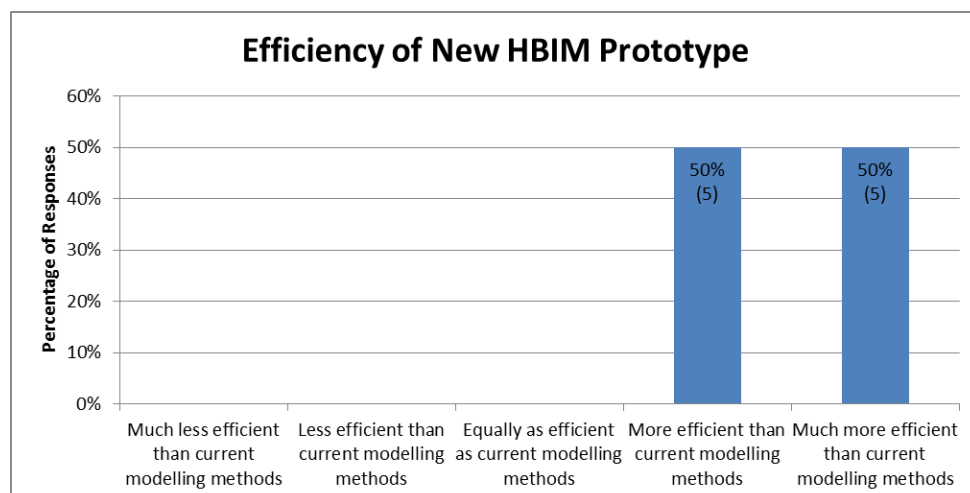


Figure 9.26: Results of end-user scenario testing – “Efficiency of New Procedural HBIM Prototypes”.

Participants were then asked to comment on the usability of the new prototypes compared to current modelling methods. 50% of respondents stated that modelling with the new prototypes is easier than current modelling methods, 40% stated that modelling with the new prototypes is much easier than current modelling methods and 10% stated that modelling with the new prototypes is equally as difficult/easy as current modelling methods (Figure 9.27).

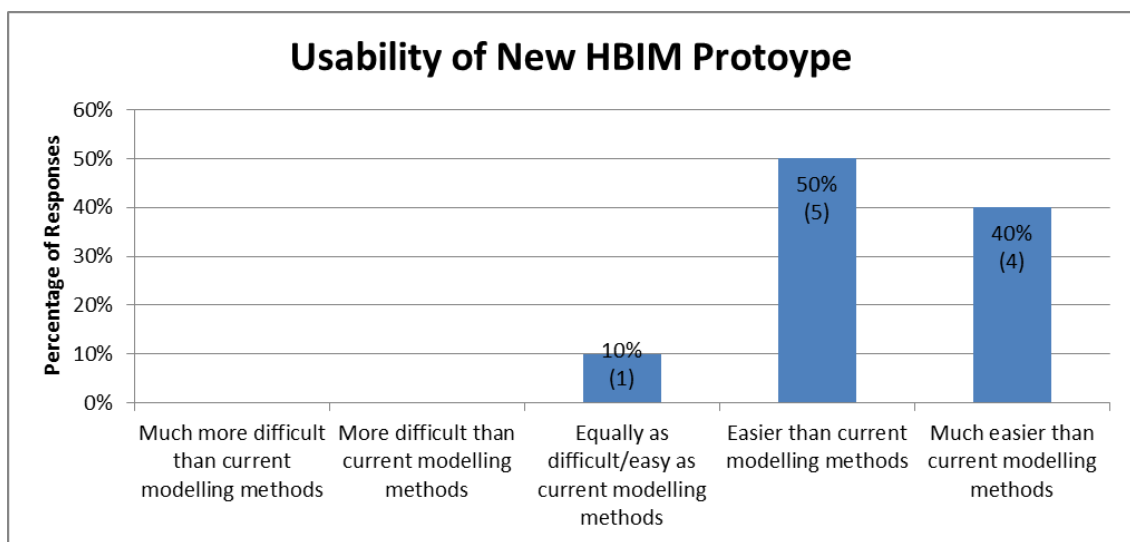


Figure 9.27: Results of end-user scenario testing – “Usability of New Procedural HBIM Prototypes”.

Participants were then asked if they find the accuracies achieved from the initial case studies with the procedural HBIM prototypes to be acceptable. 58% of respondents found the accuracy to be acceptable, 33% found the accuracy very acceptable and 8% found the accuracy somewhat acceptable (Figure 9.28).

In the third questionnaire, two additional questions were included to acquire feedback on the quality of the data and deliverables generated from the procedural HBIM prototypes. First participants were asked to comment on the suitability of HBIM deliverables for typical conservation projects. 67% of respondents found the deliverables to be somewhat suitable and 33% found the deliverables to be very suitable

(Figure 9.29). Finally, participants were asked what deliverables they found most useful for conservation projects. 100% of respondents stated that 2D documentation was the most useful deliverable for conservation projects (Figure 9.30). For this question, one of the respondents stated that when their organisation has resources available for BIM they would then find BIM to be more beneficial than current 2D deliverables.

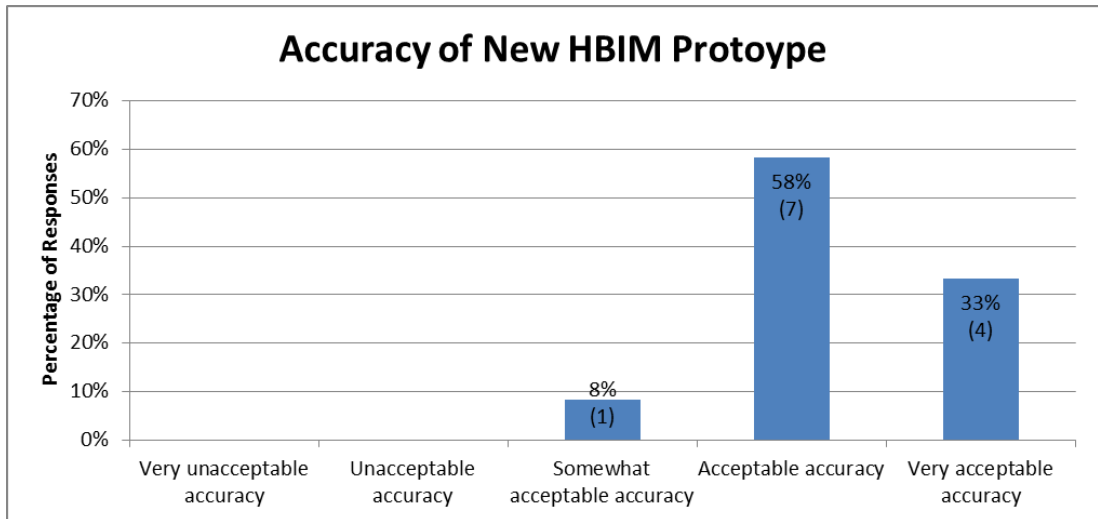


Figure 9.28: Results of end-user scenario testing – “Accuracy of New Procedural HBIM Prototypes”.

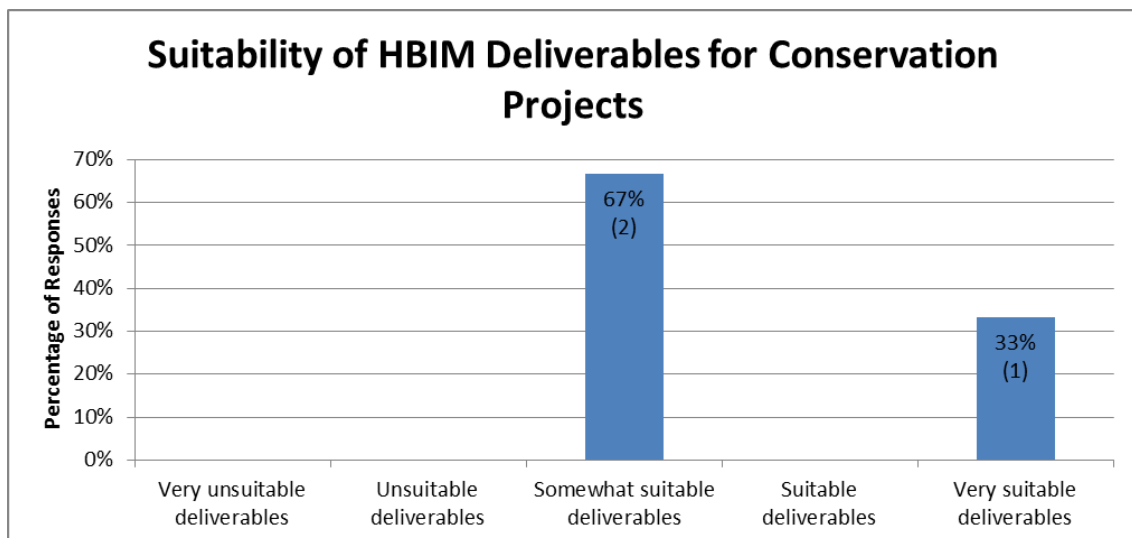


Figure 9.29: Results of end-user scenario testing – “Suitability of HBIM Deliverables for Conservation Projects”.



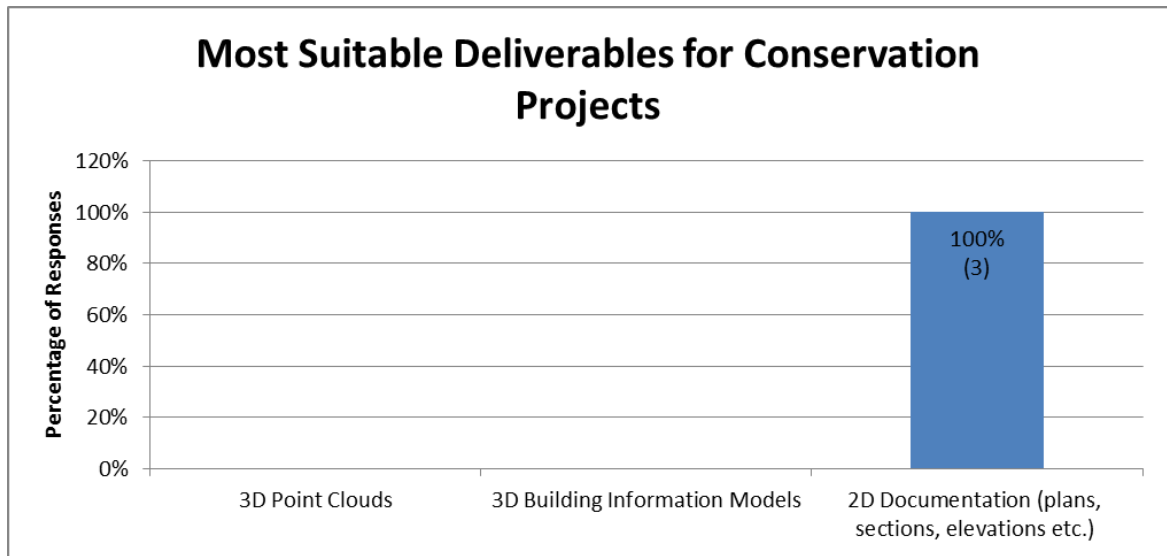


Figure 9.30: Results of end-user scenario testing – “Most Suitable Deliverables for Conservation Projects”.

At the end of each questionnaire, participants were given the opportunity to provide additional feedback. This additional feedback was optional and did not have to be answered by participants. The further feedback received from this question is shown in Table 9.4.

Table 9.4: Additional feed on procedural HBIM prototypes.

Additional Feedback on Procedural HBIM Prototypes	
Participant 1	<i>“Share via open source if possible”</i>
Participant 2	<i>“Better user-interface with graphical toolbar”</i>
Participant 3	<i>“It would be wise to combine this ArchiCAD add-on philosophy with other novel approaches currently underway, e.g. surface based object recognition, RFID and TLS, TLS to IFC.”</i>

The responses from this final part of the each questionnaire indicate that the procedural HBIM prototypes would be very useful in both industry and academia. Responses from participants suggest that the workflows with the procedural HBIM prototypes are more efficient and more usable than current manual modelling workflows. The majority of respondents were also satisfied with the accuracy capabilities of the procedural HBIM

prototypes. Additional comments from participants indicate the future prototypes could be improved with a better user-interface with a graphical toolbar. Another participant suggested that these procedural modelling concepts could be combined with other automated approaches such as surface based object recognition.

#### **9.4 Level of Automation Findings**

This section presents the findings of the level of automation test described in Chapter 8. The aim of this test was to quantify the level of automation achieved from using the HBIM procedural modelling prototypes. A test scenario was developed for this test which involved generating a building information model from survey data for a section of the Four Courts case study. This test scenario was completed with both a manual workflow and a procedural modelling workflow using the third “Irregular Procedural Building” prototype. After the test scenario was completed for both the procedural and manual workflows, it was then possible to compare and analyse the time taken to generate both models. The accuracy of both generated models was also validated against the reference point cloud.

Screen recording software was used to record the complete test scenario which can be viewed from the following link:

**[https://youtu.be/ MyJ-c2ceTQ](https://youtu.be/MyJ-c2ceTQ)**

The speed of the video has been increased for display purposes to show both the manual and procedural modelling workflows. A summary of the results from this test can be seen in Table 9.5 and Table 9.6. The formula in Equation 8.2 and Equation 8.3 were used to quantify the time saving and percentage difference in accuracy (Table 9.6).

Table 9.5: Results from Level of Automation Testing.

	<b>Manual Workflow</b>	<b>Procedural Workflow</b>
Time Taken to Generate Model	08:07.533 (minutes: seconds)	01:55.6 (minutes: seconds)
Accuracy of Generated Model	19mm Mean Error 21mm Standard Deviation	12mm Mean Error 18mm Standard Deviation

Table 9.6: Results from Level of Automation Testing - Time saving and percentage difference in accuracy.

<b>Procedural Workflow Time Saving</b>	76% Faster than manual method
<b>Procedural Workflow Accuracy Difference</b>	37% improvement in accuracy

The time taken to generate the model for the test scenario using a manual workflow was 8 minutes 7 seconds. The time taken to generate a similar model using the procedural workflow was 1 minute 55 seconds (Table 9.5). For this test scenario, the procedural modelling workflow was 76 % faster than the manual method (Table 9.6). This time saving is calculated by expressing the time difference as a percentage of the manual time.

The accuracy of the model generated with a manual workflow in its recorded time was 19mm mean error and 21mm standard deviation. The accuracy of the procedural model generated in its recorded time was 12mm mean error and 18mm standard deviation. For this test scenario using the procedural workflow resulted in a 37% improvement in accuracy. This accuracy difference was calculated by expressing the difference in accuracy as a percentage of the accuracy of the manual workflow. One reason for the improved accuracy in the procedural modelling approach was that the manual approach could only model the walls as a perfectly vertical surface. Alternatively, the procedural modelling approach was capable of generating the true condition of the wall using multiple cut-sections.

## 9.5 Discussion

The hypothesis of this research is that *“Procedural modelling is a suitable solution for more accurate, automated and easier generation of BIM geometry from point clouds”*.

In this section, the findings of the research are discussed and analysed to assess this research hypothesis.

### 9.5.1 Accuracy of Procedural HBIM Techniques

*“Procedural modelling is a suitable solution for more **ACCURATE**, automated and easier generation of BIM geometry from point clouds”.*

The accuracy of the new procedural HBIM techniques was evaluated using a quantitative deviation analysis test and by acquiring qualitative feedback from academic and industry practitioners. Feedback from participants of the end-user test indicated that a high level of accuracy is required for generating geometric building information models of existing buildings (36% required less than 1cm mean error and 27% required less than 5mm mean error). However, accuracy requirements are not the same for all projects and will vary depending on the type of project and application of the data. When participants were asked to comment on the accuracy results obtained from the Four Courts case study, 58% found the results to be acceptable and 33% found the results to be very acceptable.

The Royal Institute of Chartered Surveyors (RICS), a professional body established in the UK, provide guidelines and standard specifications for measured building surveys and BIM. The recommended accuracy requirements set out in these guidelines for measured building surveys are  $\pm 4\text{--}25\text{mm}$  depending on job specification (RICS, 2014). However, these guidelines are intended for modern architecture and not historic structures. A more relevant guide for accuracy requirements for historic buildings can

be found in the English Heritage guidelines (Bryan et al., 2009) which state that a precision of 15mm should be achieved for small structures (20m x 30m) and 25mm for large structures (40m x 60m). The procedurally generated building information model for the Four Courts case study had an overall mean error of 6mm with a standard deviation of 13mm. This tolerance is within the 15mm tolerance recommended by the English Heritage for structures of this size.

In addition to meeting this recommended accuracy requirement, the use of a procedural modelling workflow also enabled more accurate modelling than previous manual methods. The main BIM authoring software packages currently do not provide sufficient functionality for accurately modelling irregular geometry. The manual modelling of non-vertical circular walls in ArchiCAD, for example, is currently not possible with existing tools. These limitations of current BIM software result in lower accuracy models that may not accurately represent the true condition of a building. This was evident in the level of automation test results (Section 9.4) where the manually generated model with current software tools had a higher mean error (19mm) than the procedurally generated model (12mm mean error). This was due to the limited tools for modelling non-vertical circular walls in ArchiCAD. In contrast, the new procedural HBIM rules allow this type of irregular wall geometry to be accurately generated from cut-sections.

### 9.5.2 Efficiency of Procedural HBIM Techniques

*“Procedural modelling is a suitable solution for more accurate, **AUTOMATED** and easier generation of BIM geometry from point clouds”.*

The efficiency of the procedural HBIM techniques was evaluated using a combination of end-user scenario testing and a separate level of automation test scenario. During the

end-user scenario testing, users who already had experience with existing modelling approaches tested the new procedural HBIM modelling workflow. 50% of respondents stated that the new procedural HBIM workflow was much more efficient than their previous workflow while the other 50% of respondents stated that the new procedural HBIM workflow was more efficient than their previous workflow. The results from the level of automation test also indicated that a procedural modelling workflow was 76% faster than the manual workflow to create the same level of detail model.

These results indicate that the use of pre-defined procedural modelling rules can speed up the modelling process by automatically combining separate library objects to generate different parametrically controlled building arrangements. The re-use of different library objects and different parameters in procedural rules allow for very high levels of variation in the resulting models. The encoding of architectural rules and proportions into procedural modelling rules also helps to reduce the amount of further manual editing that is required. The ability to transfer survey data such as building footprints or cut-sections directly into a procedural modelling rule also greatly reduces the amount of further editing required. These capabilities of procedural modelling enable a more automated and efficient overall workflow.

### 9.5.3 Usability of Procedural HBIM Techniques

*“Procedural modelling is a suitable solution for more accurate, automated and **EASIER** generation of BIM geometry from point clouds”.*

The usability of the procedural HBIM techniques was evaluated as part of the end-user scenario test. After testing the procedural HBIM workflow, participants were asked to provide feedback on the usability of this approach compared to current modelling workflows. 50% of the respondents found the procedural HBIM workflow to be easier

than current modelling methods and 40% found the procedural HBIM workflow to be much easier than current modelling methods. These results indicate that a procedural modelling workflow is an easier and more usable approach to generating complex building geometry compared to a completely manual approach. A set of procedural rules and library objects that can be altered by parameters, enables models to be easily generated without the need for advanced 3D modelling experience.

## **9.6 Conclusion**

This chapter presented the findings and analysis of the validation and testing methodology described in Chapter 8. The results of an accuracy test, end-user scenario test and a level of automation test were presented and discussed in this chapter.

Results were presented from two accuracy tests, a physical measurement test and a deviation analysis test. The physical measurement test was carried out using the first procedural façade prototype. The results from this test which compared a generated build façade model with sampled points from a point cloud were 10mm mean error in x-coordinates, 12mm mean error in y-coordinates and 5mm mean error in z-coordinates (Table 9.3). The result from a more comprehensive deviation analysis accuracy test was 6mm (overall mean error) with a standard deviation of 13mm. This test compared all points in a reference point cloud to a procedurally generated model from the third procedural HBIM prototype. Both of these test results were within recommended accuracy guidelines from the English Heritage for buildings of this size.

The results from an end-user scenario test with academic and industry practitioners highlighted that there is a need for more accurate and automated tools for modelling existing buildings using BIM software. The responses from this end-user test also indicated that a procedural modelling workflow for generating BIM geometry from

point clouds would be of great benefit in industry and academia. Responses from participants suggest that the workflows with the procedural HBIM prototypes are more efficient and more usable than current manual modelling workflows. Respondents were also satisfied with the accuracy capabilities of the procedural HBIM prototypes.

Results were also presented in this chapter from a level of automation test. The results of this test showed that a procedural workflow was 76% faster than a manual workflow to generate a similar level of detail model. The test also showed a 37% improvement in accuracy with a procedural modelling workflow compared to a manual workflow with existing software tools.

When compared to current BIM modelling workflows, the new procedural HBIM workflow facilitated more accurate, automated and easier generation of BIM geometry from point clouds. These results showed that the research hypothesis is true and that procedural modelling is a suitable solution for improved modelling of existing buildings from point clouds.



## **Chapter Ten: Conclusions and Future Work**

### **10.1 Introduction**

In this chapter, the conclusions of the research are summarised and presented. The aim and objectives are presented next to review the success of the research in achieving these. The contributions to knowledge made by this research are then discussed followed by a review of limitations and recommendations for future work to further develop research in this area.

### **10.2 Conclusions**

The aim of this research was to assess if procedural modelling is a suitable solution for more accurate, automated and easier generation of BIM geometry from point clouds. In order to test the research hypothesis, new procedural modelling prototypes were designed, implemented and tested for reconstructing BIM geometry from point clouds. The results of an accuracy test, end-user scenario test and level of automation test proved that procedural modelling is a suitable solution for more accurate, automated and easier generation of BIM geometry from point clouds.

The main conclusions reached after designing, implementing and testing the procedural modelling prototypes for BIM software are presented below.

1. Procedural modelling can improve the efficiency, accuracy and usability of modelling existing buildings from point cloud data.
2. Architectural rules can be encoded into procedural modelling rules to speed up the reconstruction process.
3. Survey data can be transferred directly into procedural modelling rules as parameters to reduce the amount of further editing required.

4. Procedural modelling can be used for accurate modelling of irregular and complex geometries.
5. Procedural modelling rules can be combined with libraries of detailed parametric objects.
6. Parameters of procedural modelling rules can be interactively edited to instantly update the geometry of a model.
7. Procedural modelling rules can be used to control parameters of multiple library objects simultaneously.
8. Parametric scripting languages such as the Geometric Descriptive Language (GDL) can be used to implement procedural modelling rules.

The main objectives set out at the start of this research have all been met. These objectives are presented below.

- Workflows and tools were developed for combining captured survey data in a BIM environment (Section 6.2).
- A library of reusable parametric shapes and objects were designed for reconstructing building geometry from point clouds (Section 3.4.3).
- New procedural rules were designed for reconstructing building geometry from point clouds (Section 3.4, Section 4.3 and Section 5.3).
- New methods were designed for interactive editing of procedural models (Chapter 6).
- The conceptual design for a new library of shapes, procedural rules and methods for interactive editing were implemented as prototype plug-ins to existing BIM software (Section 3.5, Section 4.4 and Section 5.4).

- The developed prototypes were further implemented with real case studies (Chapter 7).
- Methods for testing and validating the developed prototypes were designed and implemented (Chapter 8).
- Results of testing and validation were analysed and used to further develop the prototype plug-ins (Chapter 9).

Based on the above analysis, the specified aim and objectives have been fulfilled by the work outlined in this thesis and the implemented software prototypes.

### **10.3 Summary of Contributions**

Throughout the work undertaken for this research, several contributions to knowledge have been made in the area of virtual building reconstruction from unstructured reality capture data. These contributions to knowledge include:

1. An extensive review of existing research into the reconstruction of BIM geometry from point clouds.
2. New tools for accurately importing survey datasets into a BIM environment.
3. New parametric shapes and objects for reconstructing BIM geometry from point clouds.
4. New procedural modelling rules and algorithms developed as plug-ins to existing BIM software. This includes procedural rules for:
  - a) Automatically generating standard vertical wall objects from floor plans.
  - b) Automatically generating non-vertical wall objects from multiple cut-sections.
  - c) Automatically splitting linear and curved façades and walls into floors and tiles.

- Automatically generating and positioning architectural objects on façade/building tiles.
5. Tools for interactive editing of computer-generated geometry. This includes efficient tools for smart editing of objects simultaneously in groups or individually.

## **10.4 Limitations and Recommendations for Further Research**

This section highlights a number of limitations with the current implementation of procedural modelling prototypes along with recommended solutions for future development. Other areas for future work relevant to this research are also presented.

### **10.4.1 Software Prototype Limitations & Recommended Solutions**

One limitation of using the Geometric Description Language for implementing procedural rules with multiple library objects is that a single GDL script can have only one semantic IFC classification. If more than one object is coded in a single script then they must have the same semantic class. It is possible to store library objects in separate scripts that have different semantic classes but if these are called as macro objects from another script then the semantic class of the calling script is assigned to all called macro scripts. This results in an incorrect semantic classification of individual library parts.

A solution to this for future development would be to only implement the library objects using the Geometric Descriptive Language (GDL) and use another language for implementing the procedural rules which combine individual library objects. As a plug-in to the ArchiCAD BIM software, all procedural rules could be implemented using C++ and the software API which would call individual library objects stored as separate GDL objects. This approach would ensure that the intended semantic classes of library objects are maintained.

Another limitation of the current implementation of procedural HBIM prototypes is that library objects are not easily interchangeable by a user. The ideal solution would be to have procedural rules completely independent of the library objects. This would enable the procedural rules to be used with many different libraries of objects that could be extended at any time. The current implementation of the prototypes does not facilitate interchangeable library objects. This is due to the fact that the current implementation of procedural rules requires library objects to be set up with parameters for multiple instances. This implementation would require custom objects to be built specifically for use with the developed procedural rules.

It is recommended for future implementations that a standardised set of parameters are used for all library objects and procedural rules. Having clearly defined standard input and output parameters for procedural rules would enable better interoperability of procedural rules with library objects. If library objects do not contain the same parameter names as required by a procedural rule, then the software should facilitate a mapping of library object parameters to rules parameters. This would facilitate the use of procedural rules with new custom library objects not defined in the software plug-in. Interchangeable library objects would also require library objects to be separate from the procedural rules. A new implementation design using C++ and the software API for all procedural rules and GDL for library objects would facilitate the use of interchangeable objects.

A final limitation of the current implementation of procedural modelling prototypes is that the large number of calculations involved in the procedural rules can lead to slow model generation and editing times. This issue of performance is only evident in the third procedural modelling prototype when a large number of cut-sections are used as inputs into the procedural rules. A number of efforts have been made to improve

performance issues such as optimising the algorithms to avoid all unnecessary calculations when altering or generating geometry. Further optimisation is recommended for future development to enable fast geometry generation and instant graphical editing.

#### **10.4.2 Recommendations for Further Research**

Future research work for advancing the newly developed concept of Procedural HBIM is proposed in the following areas:

1. Improved Software Implementations:
  - a) Improved implementation for a plug-in to ArchiCAD software to enable correct semantic classification of objects, interchangeable libraries of parametric objects, better optimisation of algorithms and a more user-friendly graphical user-interface.
  - b) Implement the conceptual design of procedural HBIM as plug-ins to other BIM and CAD authoring software platforms.
  - c) Implement the conceptual design of Procedural HBIM as a standalone desktop platform that communicates to other BIM authoring platforms via relevant software API's. The connection to other BIM software platforms should be a live connection to allow real-time update of model changes in both software packages. This standalone software platform should support point cloud imports and the export of generated models in IFC format.
  - d) Implement the conceptual design of Procedural HBIM as a web-based solution built on the open source BIMserver platform (van Berlo et al., 2016). The open source BIMserver is an object-relational database for IFC data with graphical user interfaces for accessing IFC data via a web-browser. The concepts of Procedural HBIM could be developed as a

plug-in to current BIMserver applications or alternatively, as a web-service that performs functions and communicates with BIMserver applications.

2. Develop solutions for automatic feature detection from point clouds that could be integrated with the developed procedural HBIM approach. The combination of automatic point cloud feature detection and procedural modelling has potential for achieving full automation of building reconstructions from point clouds. Successful automatic feature detection from point clouds would reduce the need for refinement of procedural models. The addition of procedural modelling would enable basic extracted elements to be automatically converted into parametric, complex and information-rich model geometry.
3. Investigate the integration of procedural HBIM concepts with 3D geographic information systems (GIS) for developing HBIM in the wider city modelling domain.
4. Investigate more interoperable methods of sharing parametric content between different software platforms that maintain parametric behaviour. One possible solution is with the use of BIMscript (BIMobject, 2016). BIMscript is an open scripting language which can be combined with a BIM object authoring software for converting scripted parametric content into native formats for various BIM authoring software such as ArchiCAD, Revit and SketchUp.
5. Develop new libraries of parametric objects for other architectural styles (e.g. gothic architecture) for use with existing procedural modelling rules.

This research has furthered the knowledge in the area of automatic building reconstruction from unstructured reality capture data. It has shown the benefit and importance that a procedural modelling approach has for improving the process of reconstructing historical building geometry from point clouds.

## REFERENCES

- Akca, D. 'Full Automatic Registration of Laser Scanner Point Clouds', *Optical 3D Measurement Techniques, VI*, Zurich Switzerland, 22-25 September 2003, 330-337.
- Allen, P. K., Troccoli, A., Smith, B., Murray, S., Stamos, I. and Leordeanu, M. (2003) 'New Methods for Digital Modeling of Historic Sites', *IEEE Comput. Graph. Appl.*, 23(6), pp. 32-41.
- Anil, E. B., Tang, P., Akinci, B. and Huber, D. 'Assessment of the quality of as-is building information models generated from point clouds using deviation analysis'. 78640F-78640F-13.
- Anil, E. B., Tang, P., Akinci, B. and Huber, D. (2013) 'Deviation analysis method for the assessment of the quality of the as-is Building Information Models generated from point cloud data', *Automation in Construction*, 35(0), pp. 507-516.
- Baik, A., Alitany, A., Boehm, J. and Robson, S. (2014) 'Jeddah Historical Building Information Modelling "JHBIM" – Object Library', *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, II-5, pp. 41-47.
- Balachowski, D. J. (2005) *HABS Guidelines, Recording Historic Structures, and Sites with HABS Measured Drawings*.
- Barazzetti, L., Scaioni, M. and Remondino, F. (2010) 'Orientation and 3D modelling from markerless terrestrial images: combining accuracy with automation', *The Photogrammetric Record*, 25(132), pp. 356--381.
- Barazzetti, L., Banfi, F., Brumana, R., Gusmeroli, G., Oreni, D., Previtali, M., Roncoroni, F. and Schiantarelli, G. (2015) 'BIM from Laser Clouds and Finite Element Analysis: Combining Structural Analysis And Geometric Complexity', *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XL-5/W4, pp. 345-350.
- Barber, D. and Mills, J. (2007) *3D Laser Scanning for Heritage: Advice and guidance to users on laser scanning in archaeology and architecture*: English Heritage.
- Barbosa, M. J., Pauwels, P., Ferreira, V. and Mateus, L (2016) "Towards increased BIM usage for existing building interventions", *Structural Survey*, Vol. 34 Issue: 2, pp.168-190
- Beacham, R., Niccolucci, F. and Denard, H. (2009): *The London Charter*. London. Available at: <http://www.londoncharter.org/>.
- Becker, S. and Haala, N. 'Grammar Supported Facade Reconstruction from Mobile Lidar Mapping '. *CMRT09 - CityModels, Roads and Traffic*, Paris, France, 3 - 4 Spetember 2009: The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science.



- Becker, S., Peter, M. and Fritsch, D. (2015) 'Grammar-Supported 3D Indoor Reconstruction From Point Clouds For "As-Built" BIM', *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, II-3/W4, pp. 17-24.
- Beraldin, J. A. (2004) 'Integration of Laser Scanning and Close-Range Photogrammetry - Last Decade and Beyond', *XXth International Society for Photogrammetry and Remote Sensing (ISPRS) Congress Istanbul, Turkey, July 12-23*, 972-983.
- Berlo, L. v. and Laat, R. d. 'Integration of BIM and GIS: The Development of the CityGML GeobBIM Extension', *5th International 3D GeoInfo Conference, November 3-4, 2010*, Berlin, Germany, Berlin, Germany.
- Bernardini, F. and Rushmeier, H. (2002) 'The 3D Model Acquisition Pipeline', *Computer Graphics Forum*, 21(2), pp. 149-172.
- BIMObject (2016) *BIMscript* & *LENA*. Available at: <http://info.bimobject.com/bimscript> 2016).
- Boehler, W. and Heinz, G. 'Documentation, Surveying, Photogrammetry', *XVII CIPA Symposium, Olinda, Brazil, 3rd - 6th October 1999*.
- Boeykens, S. (2011) 'Using 3D Design software, BIM and game engines for architectural historical reconstruction'. *CAAD Futures 2011*, Liège, Belgium, 6-8 July 2011.
- Boeykens, S., Himpe, C. and Martens, B. (2012) 'A Case Study of Using BIM in Historical Reconstruction - The Vinohrady synagogue in Prague'. *The 30th International Conference on Education and research in Computing Aided Architectural Design in Europe*, Prague, Czech Republic, 12-14 September 2012.
- Boloix, G. (1997) 'System evaluation and quality improvement', *J. Syst. Softw.*, 36(3), pp. 297-311.
- Boloix, G. and Robillard, P. N. (1995) 'A software system evaluation framework', *Computer*, 28(12), pp. 17-26.
- Borenstein, D. (1998) 'Towards a practical method to validate decision support systems', *Decision Support Systems*, 23(3), pp. 227-239.
- Bryan, P., Blake, B., Bedford, J., Barber, D., Mills, J. and David, A. (2009) *Metric Survey Specifications for Cultural Heritage*: English Heritage.
- BuildingSMART International (2013), *Industry Foundation Classes (IFC) for Data Sharing in the Construction and Facility Management Industries* (ISO 16739:2013).
- BuildingSMART Alliance (2011), *Construction Operations Building information exchange (COBie)*. Available at: [https://www.nibs.org/?page=bsa\\_cobie](https://www.nibs.org/?page=bsa_cobie)

- Byrne, C. (2015) 'Restored Four Courts dome set for seal of approval', *The Irish Times*, 7th December 2015. Available at: <http://www.irishtimes.com/news/ireland/irish-news/major-domo-1.2456337>.
- Carroll, J. M. (2000) *Making use: scenario based design of human-computer interactions*. Cambridge: The MIT Press.
- Cashman, J. 1922a. The Four Courts Bombardment (1922). *In: RTÉ Cashman Collection*.
- Cashman, J. 1922b. The Four Courts Damage (1922). *In: RTÉ Cashman Collection*.
- Centofanti, M., Continenza, R., Brusaporci, S. and Trizio, I. 'The architectural information system SIARCH3D-UNIVAQ for analysis and preservation of architectural heritage', *4th ISPRS International Workshop 3D-ARCH 2011: "3D Virtual Reconstruction and Visualization of Complex Architectures"*, Trento, Italy, 2-4 March 2011, 9-14.
- Chevrier, C., Charbonneau, N., Grussenmeyer, P. and Perrin, J.-P. (2010) 'Parametric Documenting of Built Heritage: 3D Virtual Reconstruction of Architectural Details', *International Journal of Architectural Computing*, 08(02), pp. 131-145.
- CIPA (2010) *The International Committee for Documentation of Cultural Heritage* Heritage Documentation. Available at: <http://cipa.icomos.org/> (Accessed: October 2015).
- De Luca, L. (2012) 'Methods, Formalisms and Tools for the Semantic-Based Surveying and Representation of Architectural Heritage', *Applied Geomatics*, (1866-9298).
- Dylla, K., Frischer, B., Mueller, P., Ulmer, A. and Haegler, S. (2010) 'Rome Reborn 2.0: A Case Study of Virtual City Reconstruction Using Procedural Modeling Techniques', in Frischer, B. (ed.) *Making history interactive: computer applications and quantitative methods in archaeology (CAA) ; proceedings of the 37th international conference, Williamsburg, Virginia, United States of America, March 22 - 26, 2009 BAR International Series*. Oxford u.a.: Archaeopress, pp. 62-66.
- Eppich, R., Chabbi, A. and Getty Conservation, I. (2007) *Recording, documentation, and information management for the conservation of heritage places : illustrated examples*. Los Angeles: Getty Conservation Institute.
- Fai, S. and Rafeiro, J. (2014) 'Establishing an Appropriate Level of Detail (LoD) for a Building Information Model (BIM) – West Block, Parliament Hill, Ottawa, Canada', *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, II-5, pp. 123-130.
- Fai, S., Graham, K., Duckworth, T., Wood, N. and Attar, R. (2011) 'Building Information Modelling and Heritage Documentation', *XXIII CIPA International Symposium*, Prague, Czech Republic, 12th-16th September.

- Gandon, J. 1700s. 18th Century Section, The Four Courts. Irish Architectural Archive, Merion Square, Dublin: Irish Architectural Archive.
- Garagnani, S. and Manferdini, A. M. (2013) 'Parametric Accuracy: Building Information Modeling Process Applied To The Cultural Heritage Preservation', *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XL-5/W1, pp. 87-92.
- Gomes, A.J.P. and Teixeira, J.C.G. (1991) Form feature modelling in a hybrid CSG/BRep scheme, *Computers & Graphics*, Volume 15, Issue 2, 1991, Pages 217-229.
- Gruen, A. (2012) 'Development and Status of Image Matching in Photogrammetry', *The Photogrammetric Record*, 27(137), pp. 36--57.
- Grussenmeyer, P. and Hanke, K. (2010) 'Cultural Heritage Applications', in Maas, G.V.a.H.-G. (ed.) *Airborne and Terrestrial Laser Scanning*: Wittles Publishing, pp. 271-272.
- Heesom, D. (2004) *An Analytical System for Space Planning on Construction Sites*. Doctoral Thesis, University of Wolverhampton, UK.
- Hichri, N., Stefani, C., Luca, L. D. and Veron, P. 'Review of the As-Built BIM Approaches', *5th International Workshop, 3D-ARCH-2013. 3D Virtual Reconstruction and Visualisation of Complex Architectures*, Trento, Italy, 107-112.
- Hohmann, B., Krispel, U., Havemann, S. and Fellner, D. 'CityFit - High-quality urban reconstructions by fitting shape grammars to images and derived textured point clouds'. *3D-ARCH 2009 - 3D Virtual Reconstruction and Visualization of Complex Architectures*, Trento, Italy, 25-28 February 2009.
- Hong, S., Jung, J., Kim, S., Cho, H., Lee, J. and Heo, J. (2015) 'Semi-automated approach to indoor mapping for 3D as-built building information modeling', *Computers, Environment and Urban Systems*, 51, pp. 34-46.
- Ibrahim, M. and krawczyk, R. 'The Level of Knowledge of CAD Objects within the Building Information Model', *CAADRIA 2004 Conference*, Seoul, South Korea.
- ICOMOS (2012) *International Council on Monuments and Sites*. Available at: <http://www.icomos.org/en> (Accessed: October 2015).
- Jung, J., Hong, S., Jeong, S., Kim, S., Cho, H., Hong, S. and Heo, J. (2014) 'Productive modeling for development of as-built BIM of existing indoor structures', *Automation in Construction*, 42(0), pp. 68-77.
- Kelly, G. and McCabe, H. (2006) 'A Survey of Procedural Techniques for City Generation', *ITB Journal*, (14), pp. 87-130.
- Kima, P., Kwon Choa, Y. and Chenb, J. (2016) 'Target-Free Automatic Registration of Point Clouds', *33<sup>rd</sup> International Symposium on Automation and Robotics in Construction (ISARC 2016)*, Auburn, USA

- Klein, L., Li, N. and Becerik-Gerber, B. (2012) 'Imaged-based verification of as-built documentation of operational buildings', *Automation in Construction*, 21(0), pp. 161-171.
- Leeuwen, J. and Wagter, H. (1997) 'Architectural Design-by-Features', in Junge, R. (ed.) *CAAD futures 1997*: Springer Netherlands, pp. 97-115.
- Leica Geosystems. (2009) *Leica TPS1200+ Series High performance Total Station*. Available at: [http://www.leica-geosystems.com/downloads123/zz/tps/tps1200/brochures/Leica\\_TPS1200+\\_brochure\\_en.pdf](http://www.leica-geosystems.com/downloads123/zz/tps/tps1200/brochures/Leica_TPS1200+_brochure_en.pdf).
- Liu, A., Wang, X., Wright, G., Cheng, J., Li, X. and Liu, R. (2017) *A State-of-the-Art Review on the Integration of Building Information Modeling (BIM) and Geographic Information System (GIS)*. ISPRS International Journal of Geo-Information, Vol. 6 Issue: 2, pp.53.
- Lipp, M., Wonka, P. and Wimmer, M. (2008) 'Interactive Visual editing of Grammars for Procedural Architecture', *ACM Transactions on Graphics*, 27(3), pp. 1.
- Lynch, E. M., Matthews, N. A. and Noble, T. A. (2017) 'Unraveling the enigma of prehistoric bedrock ground stone features on the Chaquaqua Plateau, using close-range photogrammetry', *Quaternary International*, Volume 439, Part B, 2 June 2017, Pages 50-68, ISSN 1040-6182.
- Matthews, N. A. (2008) *Aerial and Close-Range Photogrammetric Technology: Providing Resource Documentation, Interpretation, and Preservation*. , Denver, Colorado 80225: U.S. Department of the Interior, Bureau of Land Management, National Operations Center
- Moore, N. (2013) *A Model Based System For Contextual On-Site Construction Planning In Augmented Reality*. Doctor of Philosophy, University of Wolverhampton, Wolverhampton.
- Murphy, M. (2012) *Historic Building Information Modelling (HBIM), For Recording and Documenting Classical Architecture in Dublin 1700 to 1830*. Doctor of Philosophy, Trinity College Dublin, Dublin.
- Murphy, M., Govern, E. M. and Pavia, S. (2013) 'Historic Building Information Modelling – Adding intelligence to laser and image based surveys of European classical architecture', *ISPRS Journal of Photogrammetry and Remote Sensing*, ISSN 0924-2716.
- Müller, P., Wonka, P., Haegler, S., Ulmer, A. and Gool, L. V. (2006) 'Procedural Modeling of Buildings', *ACM Transactions on Graphics*, 25, pp. 614-623.
- Müller, P., Zeng, G., Wonka, P. and Gool, L. V. (2007) 'Image-based Procedural Modeling of Facades', *ACM Trans. Graph.*, 26(ACM), pp. 85.

- Oreni, D., Brumana, R., Della Torre, S., Banfi, F., Barazzetti, L. and Previtali, M. (2014) 'Survey turned into HBIM: the restoration and the work involved concerning the Basilica di Collemaggio after the earthquake (L'Aquila)', *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, II-5, pp. 267-273.
- Pauwels, P., Verstraeten, R., Meyer, R. D. and Campenhout, J. V. (2008) *Architectural Information Modelling for Virtual Heritage Application. Digital Heritage -- Proceedings of the 14th International Conference on Virtual Systems and Multimedia: Archaeolingua*, p. 18-23.
- Previtali, M., Barazzetti, L., Brumana, R. and Scaioni, M. (2014) 'Towards automatic indoor reconstruction of cluttered building rooms from point clouds', *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, II-5, pp. 281-288.
- Pu, S. and Vosselman, G. (2009) 'Knowledge based reconstruction of building models from terrestrial laser scanning data', *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(6), pp. 575 - 584.
- Quattrini, R., Malinverni, E. S., Clini, P., Nespeca, R. and Orlietti, E. (2015) 'From TLS to HBIM. High Quality Semantically-Aware 3D Modeling of Complex Architecture', *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XL-5/W4, pp. 367-374.
- RealSIM (2013) *3D Simulation solutions for the real world*. Available at: [www.realsim.ie](http://www.realsim.ie) (2013).
- Roche, N. (1999) *The Legacy of Light, A History of Irish Windows* Bray, Co. Wicklow: Wordwell Ltd., p. 14.
- Royal Institute of Chartered Surveyors (2014): *Measured Surveys of Land, Buildings and Utilities, 3<sup>rd</sup> Edition*. London, UK: Royal Institution of Chartered Surveyors (RICS).
- Shah, J.J. and Mantyla, M. (1995) *Parametric and Feature Based CAD/CAM*: John Wiley & Sons, Inc.
- Stiny, G. and Gips, J. (1972) 'Shape Grammars and the Generative Specification of Painting and Sculpture', *The Best Computer Papers of 1971*, pp. 125-135.
- Sun, J., Zhang, J. and Zhang, G. (2016) 'An automatic 3D point cloud registration method based on regional curvature maps', *Image and Vision Computing*, Volume 56, December 2016, Pages 49-58.
- Tang, P., Huber, D., Akinci, B., Lipmand, R. and Lytle, A. (2010) 'Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques', *Automation in Construction*, 19(7), pp. 829-843.
- Teijgeler, R. (2017) *What is Cultural Heritage*. Available at: [http://www.cultureindevelopment.nl/cultural\\_heritage/what\\_is\\_cultural\\_heritage](http://www.cultureindevelopment.nl/cultural_heritage/what_is_cultural_heritage) (Accessed: June 2017).

- Thaller, W., Krispel, U., Havemann, S., Redi, I., Redi, A. and Fellner, D. W. 'Developing Parametric Building Models - The Gandis Use Case'. *4th International Workshop 3D-ARCH 2011, "3D Virtual Reconstruction and Visualisation of Complex Architectures"*, Trento, Italy, 2-4 March.
- Theiler, P. W. and Schindler, K. (2012) 'Automatic Registration of Terrestrial Laser Scanner Point Clouds using Natural Planar Surfaces', *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, I-3, pp. 173-178.
- Thomson, C. and Boehm, J. (2015) 'Automatic Geometry Generation from Point Clouds for BIM', *Remote Sensing*, 7(9), pp. 11753.
- UMass Amherst Center (2017) *Center for Heritage and Society*. Available at: <https://www.umass.edu/chs/about/whatisheritage.html> (Accessed: June 2017).
- UNESCO (2011) *World Heritage Centre*. Available at: <http://whc.unesco.org/> (Accessed: October 2015).
- Van Berlo, L., Laat, R. d. and Beetz, J. (2016) *BIMserver*. Available at: <http://bimserver.org/> (2016).
- Volk, R., Stengel, J. and Schultmann, F. (2014) 'Building Information Modeling (BIM) for existing buildings — Literature review and future needs', *Automation in Construction*, 38(0), pp. 109-127.
- Wang, C., Cho, Y. K. and Kim, C. (2015) 'Automatic BIM component extraction from point clouds of existing buildings for sustainability applications', *Automation in Construction*, 56, pp. 1-13.
- Watson, A. (2009) *GDL Handbook, A comprehensive Guide to Creating Powerful ArchiCAD Objects*. Auckland, New Zealand: Cadimage Tools Ltd.
- Wu, T. C., Lin, Y. C., Hsu, M. F., Zheng, N. W. and Chen, W. L. (2013) 'Improving Traditional Building Repair Construction Quality using Historic Building Information Modeling Concept', *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XL-5/W2, pp. 691-694.
- Xiong, X., Adan, A., Akinci, B. and Huber, D. (2013) 'Automatic creation of semantically rich 3D building models from laser scanner data', *Automation in Construction*, 31(0), pp. 325-337.
- Zhang, R. and Zakhori, A. 'Automatic identification of window regions on indoor point clouds using LiDAR and cameras'. *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, 24-26 March 2014, 107-114.

## APPENDIX A

### Appendix A: Information Sheet for Participants of End-User Scenario Test

#### Information Sheet for Participants

##### 1. Project Details:

The main aim of this research is to develop a more automated and efficient solution for generating accurate digital 3D Building Information Models of existing buildings from survey data. A prototype plug-in to existing Building Information Modelling (BIM) software has been developed for semi-automatic modelling of existing buildings from survey data. This end-user scenario test attempts to collect information in order to assess the usability, efficiency and quality of the new software plug-in.

##### 2. End-User Test:

Participants taking part in the end-user test will first be given a short presentation to explain the project and the test procedure. Next a demonstration of the new software tools will be made to participants. After this participants will be given an opportunity to try and test the new software tools to carry out a typical end-use scenario of the software. Finally participants will be asked to complete an online questionnaire to provide feedback and to evaluate the newly developed software tools. The total amount of time required to take part in this end-user test is roughly thirty minutes. This includes a ten minutes for the initial presentation, five minutes for a demonstration of the software, ten minutes to try and test the software and five minutes for completing an online questionnaire.

##### 3. Confidentiality Statement:

All data obtained from participants will be kept confidential and will only be reported or published in aggregate format (by reporting only combined results and never reporting individual ones). All questionnaires will be concealed and no one other than the primary investigator will have access to them. The data gathered will be stored on a password protected DIT encrypted computer and backed up on a password protected DIT Google Drive account. All survey data collected will be deleted after the project has been completed.

Please note:

- Participation in this study is completely voluntary.
- Participants have the right to withdraw at any time or refuse to participate entirely.
- Participants do not have to give a reason for withdrawing from the test.

#### **4. Contact Details**

If further information is required regarding this study please contact the principal investigator or the project supervisor using the contact details provided below.

Please note that this study has been approved by the DIT Research Ethics Committee. Any queries regarding ethics can also be made to the DIT Research Ethics Committee using their contact details below.

**Principal Investigator:**

Conor Dore,  
Postgraduate Research Student,  
Dublin Institute of Technology,  
Bolton Street, Dublin 1, Ireland  
Phone: 01-4022978  
Email: conor.dore@mydit.ie

**Project Supervisor:**

Maurice Murphy,  
Lecturer and Researcher,  
Dublin Institute of Technology,  
Bolton Street, Dublin 1, Ireland  
Email: Maurice.Murphy@dit.ie

**DIT Research Ethics Committee:**

Conor McCague,  
Graduate Research School Office,  
Dublin Institute of Technology,  
Kevin Street,  
Dublin 8  
Phone: 01-4027920  
Email: conor.mccague@dit.ie



## APPENDIX B

### Appendix B: Consent Form for End-User Scenario Test

<b>Researcher's Name:</b> CONOR DORE (use block capitals) <b>Faculty/School/Department:</b> College of Engineering and Built Environment. School of Surveying and Construction Management. <b>Title of Study:</b> “Procedural Generation of Digital 3-Dimensional Building Information Models from Survey Data.”	<b>Title:</b> MR
<b>To be completed by the:</b> <b>subject/patient/volunteer/informant/interviewee/parent/guardian</b> ( <i>delete as necessary</i> )	
3.1 Have you been fully informed/read the information sheet about this study?	YES/NO
3.2 Have you had an opportunity to ask questions and discuss this study?	YES/NO
3.3. Have you received satisfactory answers to all your questions?	YES/NO
3.4 Have you received enough information about this study and any associated health and safety implications if applicable?	YES/NO
3.5 Do you understand that you are free to withdraw from this study?	
<ul style="list-style-type: none"><li>• at any time</li><li>• without giving a reason for withdrawing</li><li>• without affecting your future relationship with the Institute</li></ul>	YES/NO
3.6 Do you agree to take part in this study the results of which are likely to be published?	YES/NO
3.7 Have you been informed that this consent form shall be kept in the confidence of the researcher?	YES/NO
Signed _____ Date _____	
Name in Block Letters _____	
Signature of Researcher _____ Date _____	

## APPENDIX C

### Appendix C: End-User Scenario Questionnaire 1

#### Building Information Modelling (BIM) for Existing Buildings - Questionnaire

Dear colleague, thank you for taking the time to complete this online questionnaire.

It will take approximately 5 minutes to complete. Your participation in this study is completely voluntary. There are no risks to you, or further commitments from you, as a result of taking part in this questionnaire. All data obtained from participants will be kept confidential and will only be reported or published in aggregate format (by reporting only combined results and never reporting individual ones). You have the right to withdraw at any time or refuse to participate entirely.

Your views are very important to this research so your help will be greatly appreciated.

If you have any queries at any time while completing the questionnaire please do not hesitate to contact the principle investigator Conor Dore at 087-2125573 or by email to [conor.dore@mydit.ie](mailto:conor.dore@mydit.ie).

1. Please identify the core business activity of your organisation.

*Mark only one oval.*

- ☐ Surveying
- ☐ Architecture
- ☐ Engineering
- ☐ Construction Management
- ☐ 3D Modelling/Visualisation
- ☐ Academia
- ☐ Other:

2. How much experience do you/your company have with generating Building Information Models (BIMs) for existing buildings?

*Mark only one oval.*

- ☐ None
- ☐ Some (1 - 2 projects)
- ☐ Moderate (3 projects or more)
- ☐ Frequent (large number of projects)

3. Which of the following software do you/your company mostly use for modelling existing buildings?

*Mark only one oval.*

- ☐ Revit (Autodesk)
- ☐ ArchiCAD (Graphisoft)
- ☐ Bentley Architecture (Bentley)
- ☐ Allplan Architecture (Nemetschek)
- ☐ SketchUp (Trimble)
- ☐ 3D Studio Max (Autodesk)
- ☐ AutoCAD/AutoCAD Civil 3D (Autodesk)
- ☐ Other: .....

4. Which methods do you/your company mostly use for collecting data to model existing buildings?

*Mark only one oval.*

- ☐ Laser scanning
- ☐ Photogrammetry
- ☐ Total station surveys
- ☐ Combination of methods
- ☐ Existing 2D documentation (plans, elevations, sections, historical documents etc.)
- ☐ Other: .....

5. What would be the typical accuracy requirements for BIM projects undertaken by you/your company?

(accuracy of final BIM)

*Mark only one oval.*

- ☐ Less than 5cm mean error
- ☐ Less than 2cm mean error
- ☐ Less than 1cm mean error
- ☐ Less than 5mm mean error
- ☐ Other: .....

#### Assessment of Current BIM Software:

---

6. Do you think current BIM software provides sufficient tools for accurately modelling existing buildings? (E.g. modelling deformation, irregular geometries etc.)

*Mark only one oval.*

- ☐ Very insufficient tools for modelling existing buildings
- ☐ Insufficient tools for modelling existing buildings
- ☐ Somewhat sufficient tools for modelling existing buildings
- ☐ Sufficient tools for modelling existing buildings
- ☐ Very sufficient tools for modelling existing buildings

7. How efficient are the current software tools that you/your company use for modelling existing buildings?

*Mark only one oval.*

- ☐ Very inefficient
- ☐ Inefficient
- ☐ Somewhat efficient
- ☐ Efficient
- ☐ Very efficient

8.What do you/your company find to be the greatest challenge when modelling existing buildings with BIM?

*Mark only one oval.*

- ☐ Working with point clouds within BIM software
- ☐ Lack of suitable parametric library objects/Revit families for existing buildings
- ☐ Manual creation of custom parametric objects/Revit families
- ☐ Modelling true condition of buildings (e.g. modelling deformation, irregular objects)
- ☐ Lack of tools for checking accuracy/quality control
- ☐ Other:

9.Do you/your company currently use any automation in your workflow for modelling existing buildings?

*Mark only one oval.*

- ☐ Yes
- ☐ No

10.If answered yes to previous question please specify the type of automation used (e.g automatic detection of planes, features etc.) and software used for automation.

.....

.....

.....

.....

11. Please indicate if you think there is a need for more automation in the current workflows for modelling existing buildings from point clouds.

*Mark only one oval.*

- ☐ Strongly disagree
- ☐ Disagree
- ☐ Neutral
- ☐ Agree
- ☐ Strongly agree

12. Please rate the usefulness of a procedural modelling approach for BIM (different geometric arrangements automatically generated from pre-defined rules and parametric library objects).

*Mark only one oval.*

- ☐ Not at all useful
- ☐ Not very useful
- ☐ Somewhat useful
- ☐ Useful
- ☐ Very useful

### Assessment of New HBIM Prototype Plug-in for ArchiCAD BIM Software:

---

13. Please rate the usefulness of the new HBIM prototype for modelling existing buildings with BIM.

*Mark only one oval.*

- ☐ Not at all useful
- ☐ Not very useful
- ☐ Somewhat useful
- ☐ Useful
- ☐ Very useful

14. Which aspects of the new prototype (if any) would you find most useful?

*Mark only one oval.*

- ☐ Automatic generation of non-vertical wall objects from sections
- ☐ Automatic generation of regular wall objects from floorplans
- ☐ Automatic splitting of buildings into floors/tiles
- ☐ Automatic generation of library objects on buildings
- ☐ Group and individual graphical editing of objects
- ☐ None

15. Please rate the efficiency of modelling with the new HBIM prototype when compared to current modelling methods.

*Mark only one oval.*

- ☐ Much less efficient than current modelling methods
- ☐ Less efficient than current modelling methods
- ☐ Equally as efficient as current modelling methods
- ☐ More efficient than current modelling methods
- ☐ Much more efficient than current modelling methods

16. Based on the demo, how easy/difficult to use do you think the new HBIM prototype is when compared to current modelling methods?

*Mark only one oval.*

- ☐ Much more difficult than current modelling methods
- ☐ More difficult than current modelling methods
- ☐ Equally as difficult/easy as current modelling methods
- ☐ Easier than current modelling methods
- ☐ Much easier than current modelling methods

17. Do you find the accuracies achieved from the initial case studies with the HBIM prototype plug-in to be acceptable?

*Mark only one oval.*

- ☐ Very unacceptable accuracy
- ☐ Unacceptable accuracy
- ☐ Somewhat acceptable accuracy
- ☐ Acceptable accuracy
- ☐ Very acceptable accuracy

18. Further feedback on HBIM prototype (Optional)

Limitations, suggestions etc.

.....

.....

.....

.....

## APPENDIX D

### Appendix D: End-User Scenario Questionnaire 2

#### Modelling Existing Buildings - Questionnaire

Dear colleague, thank you for taking the time to complete this online questionnaire.

It will take approximately 5 minutes to complete. Your participation in this study is completely voluntary. There are no risks to you, or further commitments from you, as a result of taking part in this questionnaire. All data obtained from participants will be kept confidential and will only be reported or published in aggregate format (by reporting only combined results and never reporting individual ones). You have the right to withdraw at any time or refuse to participate entirely.

Your views are very important to this research so your help will be greatly appreciated.

If you have any queries at any time while completing the questionnaire please do not hesitate to contact the principle investigator Conor Dore at 087-2125573 or by email to [conor.dore@mydit.ie](mailto:conor.dore@mydit.ie).

2. Please identify the core activity of your organisation.

*Mark only one oval.*

- ☐ 3D Modelling/Visualisation
- ☐ Academia
- ☐ Virtual/Augmented Reality
- ☐ Surveying
- ☐ Architecture
- ☐ Engineering
- ☐ Construction Management
- ☐ Other: .....

3. How much experience do you/your company have with generating digital models of existing buildings?

*Mark only one oval.*

- ☐ None
- ☐ Some (1 - 2 projects)
- ☐ Moderate (3 projects or more)
- ☐ Frequent (Large number of projects)



4. Which of the following software do you/your company mostly use for modelling existing buildings?

*Mark only one oval.*

- ☐ ArchiCAD (Graphisoft)
- ☐ Revit (Autodesk)
- ☐ SketchUp (Trimble)
- ☐ AutoCAD/AutoCAD Civil 3D (Autodesk)
- ☐ 3D Studio Max (Autodesk)
- ☐ Cinema 4D (MAXON Computer GmbH)
- ☐ Bentley Architecture (Bentley)
- ☐ Other: .....

5. Which methods do you/your company mostly use for collecting data to model existing buildings?

*Mark only one oval.*

- ☐ Laser scanning
- ☐ Photogrammetry
- ☐ Total station surveys
- ☐ Combination of methods
- ☐ Existing 2D documentation (plans, elevations, sections, historical documents etc.)
- ☐ Other: .....

6. What would be the typical accuracy requirements for modelling projects undertaken by you/your company?

*Mark only one oval.*

- ☐ Accuracy not important
- ☐ Less than 10cm mean error
- ☐ Less than 5cm mean error
- ☐ Less than 2cm mean error
- ☐ Less than 1cm mean error
- ☐ Other: .....

## Assessment of Current Modelling Software:

---

7. Do you think current modelling software provides sufficient tools for modelling existing buildings? (E.g. modelling deformation, irregular geometries etc.)

*Mark only one oval.*

- ☐ Very insufficient tools for modelling existing buildings
- ☐ Insufficient tools for modelling existing buildings
- ☐ Somewhat sufficient tools for modelling existing buildings
- ☐ Sufficient tools for modelling existing buildings
- ☐ Very sufficient tools for modelling existing buildings

8. How efficient are the current software tools that you/your company use for modelling existing buildings?

*Mark only one oval.*

- ☐ Very inefficient
- ☐ Inefficient
- ☐ Somewhat efficient
- ☐ Efficient
- ☐ Very efficient

9. What do you/your company find to be the greatest challenge when modelling existing buildings?

*Mark only one oval.*

- ☐ Working with survey data within modelling software
- ☐ Lack of suitable library objects for existing buildings
- ☐ Manual creation of custom objects for existing buildings
- ☐ Modelling true condition of buildings (e.g. modelling deformation, irregular objects)
- ☐ Lack of tools for checking accuracy/quality control
- ☐ Other: .....

10. Do you/your company currently use any automation in your workflow for modelling existing buildings?

*Mark only one oval.*

☐ Yes

☐ No

11. If answered yes to previous question please specify the type of automation used (e.g automatic detection of planes, features etc.) and software used for automation.

.....

.....

.....

.....

12. Please indicate if you think there is a need for more automation in the current workflows for modelling existing buildings from survey data.

*Mark only one oval.*

☐ Strongly disagree

☐ Disagree

☐ Neutral

☐ Agree

☐ Strongly agree

13. Please rate the usefulness of a procedural modelling approach for modelling existing buildings (different geometric arrangements automatically generated from pre-defined rules and parametric library objects).

*Mark only one oval.*

☐ Not at all useful

☐ Not very useful

☐ Somewhat useful

☐ Useful

☐ Very useful

## Assessment of New HBIM Prototype Plug-in for ArchiCAD BIM Software:

---

14. Please rate the usefulness of the new HBIM prototype for modelling existing buildings with BIM.

*Mark only one oval.*

- ☐ Not at all useful
- ☐ Not very useful
- ☐ Somewhat useful
- ☐ Useful
- ☐ Very useful

15. Which aspects of the new prototype (if any) would you find most useful?

*Mark only one oval.*

- ☐ Automatic generation of non-vertical wall objects from sections
- ☐ Automatic generation of regular wall objects from floorplans
- ☐ Automatic splitting of buildings into floors/tiles
- ☐ Automatic generation of library objects on buildings
- ☐ Group and individual graphical editing of objects
- ☐ None

16. Please rate the efficiency of modelling with the new HBIM prototype when compared to current modelling methods.

*Mark only one oval.*

- ☐ Much less efficient than current modelling methods
- ☐ Less efficient than current modelling methods
- ☐ Equally as efficient as current modelling methods
- ☐ More efficient than current modelling methods
- ☐ Much more efficient than current modelling methods

17. Based on the demo, how easy/difficult to use do you think the new HBIM prototype is when compared to current modelling methods?

*Mark only one oval.*

- ☐ Much more difficult than current modelling methods
- ☐ More difficult than current modelling methods
- ☐ Equally as difficult/easy as current modelling methods
- ☐ Easier than current modelling methods
- ☐ Much easier than current modelling methods

18. Do you find the accuracies achieved from the initial case studies with the HBIM prototype plug-in to be acceptable?

*Mark only one oval.*

- ☐ Very unacceptable accuracy
- ☐ Unacceptable accuracy
- ☐ Somewhat acceptable accuracy
- ☐ Acceptable accuracy
- ☐ Very acceptable accuracy

19. Further feedback on HBIM prototype (Optional)

Limitations, suggestions etc.

.....

.....

.....

## APPENDIX E

### Appendix E: End-User Scenario Questionnaire 3

#### Procedural HBIM Prototype - Questionnaire

Dear colleague, thank you for taking the time to complete this online questionnaire.

It will take approximately 5 minutes to complete. Your participation in this study is completely voluntary. There are no risks to you, or further commitments from you, as a result of taking part in this questionnaire. All data obtained from participants will be kept confidential and will only be reported or published in aggregate format (by reporting only combined results and never reporting individual ones). You have the right to withdraw at any time or refuse to participate entirely.

Your views are very important to this research so your help will be greatly appreciated.

If you have any queries at any time while completing the questionnaire please do not hesitate to contact the principle investigator Conor Dore at 087-2125573 or by email to [conor.dore@mydit.ie](mailto:conor.dore@mydit.ie).

3. Please identify the core activity of your organisation.

*Mark only one oval.*

- ☐ Architecture
- ☐ Surveying
- ☐ Engineering
- ☐ Conservation
- ☐ Construction Management
- ☐ Academia
- ☐ 3D Modelling/Visualisation
- ☐ Virtual/Augmented Reality
- ☐ Other: .....

4. How much experience do you/your company have with generating digital models of existing buildings?

*Mark only one oval.*

- ☐ None
- ☐ Some (1 - 2 projects)
- ☐ Moderate (3 projects or more)
- ☐ Frequent (Large number of projects)

5. Which of the following software do you/your company mostly use for modelling existing buildings?

*Mark only one oval.*

- ☐ ArchiCAD (Graphisoft)
- ☐ Revit (Autodesk)
- ☐ SketchUp (Trimble)
- ☐ AutoCAD/AutoCAD Civil 3D (Autodesk)
- ☐ 3D Studio Max (Autodesk)
- ☐ Cinema 4D (MAXON Computer GmbH)
- ☐ Bentley Architecture (Bentley)
- ☐ Other: .....

6. Which methods do you/your company mostly use for collecting data to model existing buildings?

*Mark only one oval.*

- ☐ Laser scanning
- ☐ Photogrammetry
- ☐ Total station surveys
- ☐ Combination of methods
- ☐ Existing 2D documentation (plans, elevations, sections, historical documents etc.)
- ☐ Other: .....

7. What would be the typical accuracy requirements for modelling projects undertaken by you/your company?

*Mark only one oval.*

- ☐ Accuracy not important
- ☐ Less than 10cm mean error
- ☐ Less than 5cm mean error
- ☐ Less than 2cm mean error
- ☐ Less than 1cm mean error
- ☐ Other: .....

### Assessment of Current Modelling software:

---

8. Do you think current modelling software provides sufficient tools for modelling existing buildings? (E.g. modelling deformation, irregular geometries etc.)

*Mark only one oval.*

- ☐ Very insufficient tools for modelling existing buildings
- ☐ Insufficient tools for modelling existing buildings
- ☐ Somewhat sufficient tools for modelling existing buildings
- ☐ Sufficient tools for modelling existing buildings
- ☐ Very sufficient tools for modelling existing buildings

9. How efficient are the current software tools that you/your company use for modelling existing buildings?

*Mark only one oval.*

- ☐ Very inefficient
- ☐ Inefficient
- ☐ Somewhat efficient
- ☐ Efficient
- ☐ Very efficient



10. What do you/your company find to be the greatest challenge when modelling existing buildings?

*Mark only one oval.*

- ☐ Working with survey data within modelling software
- ☐ Lack of suitable library objects for existing buildings
- ☐ Manual creation of custom objects for existing buildings
- ☐ Modelling true condition of buildings (e.g. modelling deformation, irregular objects)
- ☐ Lack of tools for checking accuracy/quality control
- ☐ Other: .....

11. Do you/your company currently use any automation in your workflow for modelling existing buildings?

*Mark only one oval.*

- ☐ Yes
- ☐ No

12. If answered yes to previous question please specify the type of automation used (e.g. automatic detection of planes, features etc.) and software used for automation.

.....

.....

.....

.....

13. Please indicate if you think there is a need for more automation in the current workflows for modelling existing buildings from survey data.

*Mark only one oval.*

- ☐ Strongly disagree
- ☐ Disagree
- ☐ Neutral
- ☐ Agree
- ☐ Strongly agree

14. Please rate the usefulness of a procedural modelling approach for modelling existing buildings (different geometric arrangements automatically generated from pre-defined rules and parametric library objects).

*Mark only one oval.*

- ☐ Not at all useful
- ☐ Not very useful
- ☐ Somewhat useful
- ☐ Useful
- ☐ Very useful

#### Assessment of New HBIM Prototype Plug-in for ArchiCAD BIM Software:

15. Please rate the usefulness of the new HBIM prototype for modelling existing buildings with BIM.

*Mark only one oval.*

- ☐ Not at all useful
- ☐ Not very useful
- ☐ Somewhat useful
- ☐ Useful
- ☐ Very useful

16. Which aspects of the new prototype (if any) would you find most useful?

*Mark only one oval.*

- ☐ Automatic generation of non-vertical wall objects from sections
- ☐ Automatic generation of regular wall objects from floorplans
- ☐ Automatic splitting of buildings into floors/tiles
- ☐ Automatic generation of library objects on buildings
- ☐ Group and individual graphical editing of objects
- ☐ None

17. Please rate the efficiency of modelling with the new HBIM prototype when compared to current modelling methods.

*Mark only one oval.*

- ☐ Much less efficient than current modelling methods
- ☐ Less efficient than current modelling methods
- ☐ Equally as efficient as current modelling methods
- ☐ More efficient than current modelling methods
- ☐ Much more efficient than current modelling methods

18. How easy/difficult to use is the new HBIM prototype when compared to current modelling methods?

*Mark only one oval.*

- ☐ Much more difficult than current modelling methods
- ☐ More difficult than current modelling methods
- ☐ Equally as difficult/easy as current modelling methods
- ☐ Easier than current modelling methods
- ☐ Much easier than current modelling methods

19. Do you find the accuracies achieved from the initial case studies with the HBIM prototype plug-in to be acceptable?

*Mark only one oval.*

- ☐ Very unacceptable accuracy
- ☐ Unacceptable accuracy
- ☐ Somewhat acceptable accuracy
- ☐ Acceptable accuracy
- ☐ Very acceptable accuracy

20. How suitable are the HBIM deliverables for typical conservation projects?

*Mark only one oval.*

- ☐ Very unsuitable deliverables
- ☐ Unsuitable deliverables
- ☐ Somewhat suitable deliverables
- ☐ Suitable deliverables
- ☐ Very suitable deliverables

19.What deliverables do you find most useful?

*Mark only one oval.*

- ☐ 3D Point Clouds
- ☐ 3D Building Information Models
- ☐ 2D Documentation (plans, sections, elevations etc.)

20.Further feedback on HBIM prototype or deliverables (Optional)  
Limitations, suggestions etc.

.....

.....

.....

.....

## Appendix F

Appendix F: Results of physical measurement accuracy test showing deviations between total station measurements and a procedurally generated building façade model for number 3 Henrietta Street.

Point ID	Total Station			BIM			Difference BIM & TS		
	x	y	z	x	y	z	x	y	z
1	2.8075	10.1753	3.2913	2.81	10.169	3.292	0.002	-0.006	0.001
2	3.4243	9.023	3.3001	3.429	9.007	3.295	0.005	-0.016	-0.005
3	2.8565	10.1972	7.9984	2.851	10.19	7.962	-0.006	-0.007	-0.036
4	3.4758	9.0387	8.0077	3.464	9.039	7.972	-0.012	0.000	-0.036
5	2.8728	10.2504	12.1848	2.876	10.23	12.128	0.003	-0.020	-0.057
6	3.5063	9.0714	12.1994	3.496	9.065	12.141	-0.010	-0.006	-0.058
7	2.892	10.2662	15.2158	2.895	10.258	15.142	0.003	-0.008	-0.074
8	3.5261	9.0835	15.2139	3.512	9.098	15.135	-0.014	0.015	-0.079
9	4.4362	7.1419	3.3084	4.427	7.132	3.315	-0.009	-0.010	0.007
10	5.0617	5.9972	3.3161	5.047	5.968	3.324	-0.015	-0.029	0.008
11	4.4784	7.169	8.0137	4.466	7.156	7.99	-0.012	-0.013	-0.024
12	5.1054	6.0054	8.0182	5.085	5.995	7.996	-0.020	-0.010	-0.022
13	4.5121	7.2156	12.2036	4.487	7.204	12.156	-0.025	-0.012	-0.048
14	5.1273	6.0437	12.2244	5.117	6.022	12.177	-0.010	-0.022	-0.047
15	4.5267	7.2322	15.2178	4.51	7.224	15.175	-0.017	-0.008	-0.043
16	5.1674	6.0556	15.2229	5.131	6.057	15.145	-0.036	0.001	-0.078
17	6.2408	3.8826	8.0245	6.225	3.853	8.015	-0.016	-0.030	-0.009
18	6.8654	2.7123	8.032	6.837	2.703	8.013	-0.028	-0.009	-0.019
19	6.2757	3.899	12.2307	6.258	3.879	12.196	-0.018	-0.020	-0.035
20	6.8865	2.7316	12.2318	6.874	2.721	12.191	-0.013	-0.011	-0.041
21	6.3046	3.9341	15.2264	6.279	3.902	15.15	-0.026	-0.032	-0.076
22	6.9159	2.7486	15.2101	6.9	2.735	15.136	-0.016	-0.014	-0.074
23	7.8698	0.76	3.329	7.831	0.739	3.346	-0.039	-0.021	0.017
24	8.4595	-0.4142	3.3516	8.434	-0.394	3.349	-0.026	0.020	-0.003
25	7.9044	0.7769	8.037	7.877	0.75	8.018	-0.027	-0.027	-0.019
26	8.5189	-0.4161	8.0451	8.512	-0.443	8.026	-0.007	-0.027	-0.019
27	7.9362	0.7759	12.2205	7.918	0.761	12.183	-0.018	-0.015	-0.037
28	8.578	-0.3931	12.2428	8.549	-0.425	12.202	-0.029	-0.032	-0.041
29	7.941	0.7994	15.21	7.938	0.786	15.153	-0.003	-0.013	-0.057
30	8.5641	-0.3886	15.2156	8.567	-0.397	15.159	0.003	-0.008	-0.057
						Average Difference	-0.015	-0.013	-0.035
						Standard Deviation	0.012	0.012	0.028
						RMSE:	0.015	0.013	0.035

## Appendix G

Appendix G: Additional check for physical measurement accuracy test to verify deviations between total station measurements and laser scan point cloud measurements.

Point ID	Total Station (TS)			Point Cloud (PC)			Difference TS & PC		
	x	y	z	x	y	z	x	y	z
1	2.8075	10.1753	3.2913	2.81	10.173	3.291	0.002	-0.002	0.000
2	3.4243	9.023	3.3001	3.431	9.003	3.299	0.007	-0.020	-0.001
3	2.8565	10.1972	7.9984	2.855	10.199	7.963	-0.002	0.002	-0.035
4	3.4758	9.0387	8.0077	3.479	9.038	7.966	0.003	-0.001	-0.042
5	2.8728	10.2504	12.1848	2.891	10.248	12.13	0.018	-0.002	-0.055
6	3.5063	9.0714	12.1994	3.512	9.075	12.142	0.006	0.004	-0.057
7	2.892	10.2662	15.2158	2.903	10.277	15.145	0.011	0.011	-0.071
8	3.5261	9.0835	15.2139	3.527	9.105	15.148	0.001	0.021	-0.066
9	4.4362	7.1419	3.3084	4.431	7.139	3.316	-0.005	-0.003	0.008
10	5.0617	5.9972	3.3161	5.055	5.97	3.321	-0.007	-0.027	0.005
11	4.4784	7.169	8.0137	4.468	7.166	7.983	-0.010	-0.003	-0.031
12	5.1054	6.0054	8.0182	5.094	5.995	7.999	-0.011	-0.010	-0.019
13	4.5121	7.2156	12.2036	4.505	7.216	12.146	-0.007	0.000	-0.058
14	5.1273	6.0437	12.2244	5.122	6.03	12.179	-0.005	-0.014	-0.045
15	4.5267	7.2322	15.2178	4.517	7.25	15.173	-0.010	0.018	-0.045
16	5.1674	6.0556	15.2229	5.15	6.064	15.155	-0.017	0.008	-0.068
17	6.2408	3.8826	8.0245	6.22	3.86	8.005	-0.021	-0.023	-0.019
18	6.8654	2.7123	8.032	6.84	2.705	8.008	-0.025	-0.007	-0.024
19	6.2757	3.899	12.2307	6.254	3.877	12.195	-0.022	-0.022	-0.036
20	6.8865	2.7316	12.2318	6.863	2.707	12.188	-0.023	-0.025	-0.044
21	6.3046	3.9341	15.2264	6.274	3.913	15.149	-0.031	-0.021	-0.077
22	6.9159	2.7486	15.2101	6.896	2.716	15.138	-0.020	-0.033	-0.072
23	7.8698	0.76	3.329	7.83	0.744	3.341	-0.040	-0.016	0.012
24	8.4595	-0.4142	3.3516	8.421	-0.451	3.353	-0.039	-0.037	0.001
25	7.9044	0.7769	8.037	7.856	0.769	8.001	-0.048	-0.008	-0.036
26	8.5189	-0.4161	8.0451	8.492	-0.464	8.028	-0.027	-0.048	-0.017
27	7.9362	0.7759	12.2205	7.9	0.753	12.179	-0.036	-0.023	-0.041
28	8.578	-0.3931	12.2428	8.537	-0.439	12.205	-0.041	-0.046	-0.038
29	7.941	0.7994	15.21	7.909	0.783	15.157	-0.032	-0.016	-0.053
30	8.5641	-0.3886	15.2156	8.552	-0.434	15.17	-0.012	-0.045	-0.046
						Average Difference	-0.015	-0.013	-0.036
						Standard Deviation	0.017	0.018	0.026
						RMSE:	0.015	0.013	0.036

## Appendix H

Appendix H: Additional check for physical measurement accuracy test to verify deviations between laser scan point cloud and a procedurally generated building façade model for number 3 Henrietta Street.

Point ID	Point Cloud (PC)			BIM			Difference PC & BIM		
	x	x	x	x	y	z	x	y	z
1	2.8075	10.1753	3.2913	2.810	10.169	3.292	0.000	0.004	-0.001
2	3.4243	9.023	3.3001	3.429	9.007	3.295	0.002	-0.004	0.004
3	2.8565	10.1972	7.9984	2.851	10.190	7.962	0.004	0.009	0.001
4	3.4758	9.0387	8.0077	3.464	9.039	7.972	0.015	-0.001	-0.006
5	2.8728	10.2504	12.1848	2.876	10.230	12.128	0.015	0.018	0.002
6	3.5063	9.0714	12.1994	3.496	9.065	12.141	0.016	0.010	0.001
7	2.892	10.2662	15.2158	2.895	10.258	15.142	0.008	0.019	0.003
8	3.5261	9.0835	15.2139	3.512	9.098	15.135	0.015	0.007	0.013
9	4.4362	7.1419	3.3084	4.427	7.132	3.315	0.004	0.007	0.001
10	5.0617	5.9972	3.3161	5.047	5.968	3.324	0.008	0.002	-0.003
11	4.4784	7.169	8.0137	4.466	7.156	7.990	0.002	0.010	-0.007
12	5.1054	6.0054	8.0182	5.085	5.995	7.996	0.009	0.000	0.003
13	4.5121	7.2156	12.2036	4.487	7.204	12.156	0.018	0.012	-0.010
14	5.1273	6.0437	12.2244	5.117	6.022	12.177	0.005	0.008	0.002
15	4.5267	7.2322	15.2178	4.510	7.224	15.175	0.007	0.026	-0.002
16	5.1674	6.0556	15.2229	5.131	6.057	15.145	0.019	0.007	0.010
17	6.2408	3.8826	8.0245	6.225	3.853	8.015	-0.005	0.007	-0.010
18	6.8654	2.7123	8.032	6.837	2.703	8.013	0.003	0.002	-0.005
19	6.2757	3.899	12.2307	6.258	3.879	12.196	-0.004	-0.002	-0.001
20	6.8865	2.7316	12.2318	6.874	2.721	12.191	-0.011	-0.014	-0.003
21	6.3046	3.9341	15.2264	6.279	3.902	15.150	-0.005	0.011	-0.001
22	6.9159	2.7486	15.2101	6.900	2.735	15.136	-0.004	-0.019	0.002
23	7.8698	0.76	3.329	7.831	0.739	3.346	-0.001	0.005	-0.005
24	8.4595	-0.4142	3.3516	8.434	-0.394	3.349	-0.013	-0.057	0.004
25	7.9044	0.7769	8.037	7.877	0.750	8.018	-0.021	0.019	-0.017
26	8.5189	-0.4161	8.0451	8.512	-0.443	8.026	-0.020	-0.021	0.002
27	7.9362	0.7759	12.2205	7.918	0.761	12.183	-0.018	-0.008	-0.004
28	8.578	-0.3931	12.2428	8.549	-0.425	12.202	-0.012	-0.014	0.003
29	7.941	0.7994	15.21	7.938	0.786	15.153	-0.029	-0.003	0.004
30	8.5641	-0.3886	15.2156	8.567	-0.397	15.159	-0.015	-0.037	0.011
						Average Difference	0.010	0.012	0.005
						Standard Deviation	0.013	0.017	0.006
						RMSE:	0.010	0.012	0.005

## **Appendix I**

Contents of Accompanying CD:

- 1) GDL source code for Procedural HBIM prototypes
- 2) C++ source code for Procedural HBIM prototypes